Semantische Technologien zur domänenspezifischen und formalen Beschreibung von Zeitreihen in Datenbanken

Ulf Noyer

Berichte aus dem DLR-Institut
für Verkehrssystemtechnik

Band 23

DLR Deutsches Zentrum
für Luft- und Raumfahrt

Semantische Technologien zur domänenspezifischen und formalen Beschreibung von Zeitreihen in Datenbanken

Von der Fakultät für Maschinenbau
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung der Würde

eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Dissertation

von:	Dipl.-Inform., M.Sc. Ulf Noyer
aus (Geburtsort):	Hildesheim

eingereicht am:	27. Mai 2013
mündliche Prüfung am:	15. November 2013

Gutachter:	Prof. Dr.-Ing. Karsten Lemmer
	Prof. Dr. Hans-Dieter Ehrich
	PD Dr. Frank Köster

2014

Berichte aus dem DLR-Institut für Verkehrssystemtechnik

Band 23

Semantische Technologien zur domänenspezifischen und formalen Beschreibung von Zeitreihen in Datenbanken

Ulf Noyer

Herausgeber:
Deutsches Zentum für Luft- und Raumfahrt e. V.
Institut für Verkehrssystemtechnik
Lilienthalplatz 7, 38108 Braunschweig

ISSN: 1866-721X

DLR-TS 1.23

Braunschweig, im Juni 2014

Institutsdirektor:
Prof. Dr.-Ing. Karsten Lemmer

Verfasser:
Ulf Noyer

Vorwort des Herausgebers

Liebe Leserinnen und Leser,

in Ihren Händen halten Sie einen Band unserer Buchreihe „Berichte aus dem DLR-Institut für Verkehrssystemtechnik". In dieser Reihe veröffentlichen wir spannende, wissenschaftliche Themen aus dem Institut für Verkehrssystemtechnik des Deutschen Zentrums für Luft- und Raumfahrt e.V. (DLR) und aus seinem Umfeld. Einen Teil der Auflage stellen wir Bibliotheken und Fachbibliotheken für ihren Buchbestand zur Verfügung. Herausragende wissenschaftliche Arbeiten und Dissertationen finden hier ebenso Platz wie Projektberichte und Beiträge zu Tagungen in unserem Hause von verschiedenen Referenten aus Wirtschaft, Wissenschaft und Politik.

Mit dieser Veröffentlichungsreihe verfolgen wir das Ziel, einen weiteren Zugang zu wissenschaftlichen Arbeiten und Ergebnissen zu ermöglichen. Wir nutzen die Reihe auch als praktische Nachwuchsförderung durch die Publikation der wissenschaftlichen Ergebnisse von Dissertationen unserer Mitarbeiter und auch externer Doktoranden. Veröffentlichungen sind wichtige Meilensteine auf dem akademischen Berufsweg. Mit der Reihe „Berichte aus dem DLR-Institut für Verkehrssystemtechnik" erweitern wir das Spektrum der möglichen Publikationen um einen Baustein. Darüber hinaus verstehen wir die Kommunikation unserer Forschungsthemen als Beitrag zur nationalen und internationalen Forschungslandschaft auf den Gebieten Automotive, Bahnsysteme und Verkehrsmanagement.

Bei Fahrversuchen für die Entwicklung und Erprobung von Assistenz- und Automationssystemen fallen große Datenmengen in Form von Messreihen an. Die Interpretation der Versuchsergebnisse stellt insofern eine Herausforderung dar, als dass diese Zahlenreihen mit Fachbegriffen und Modellen aus der Anwendungsdomäne in Verbindung gesetzt werden müssen. Der vorliegende Band stellt einen Ansatz vor, der es ermöglicht, in Datenbanken gespeicherte Messdaten mit formalen Modellen zur Beschreibung zu verknüpfen. Kern der Methodik sind Annotationen, welche als verbindendes Element eine Brücke zwischen den Daten zum einen und den Modellen zum anderen bilden. Die vorgestellte Methode erlaubt es, für die Daten modellbasierte Verfahren anzuwenden und von deren Vorteilen zu profitieren. So lassen sich beispielsweise Modelle wiederverwenden und formale Methoden einsetzen, um die erzielten Ergebnisse schließlich wieder mit den Daten in Beziehung zu setzen. Die vorgestellte Dissertation liefert somit einen Beitrag zur ausdrucksstarken und formalen Beschreibung und Analyse von Messdaten, wie sie beispielsweise bei der Durchführung von Experimenten und empirischen Studien in großen Mengen gesammelt werden.

Prof. Dr.-Ing. Karsten Lemmer

Vorwort des Autors

Diese Dissertation ist begleitend zu meiner wissenschaftlichen Tätigkeit am Institut für Verkehrssystemtechnik des Deutschen Zentrums für Luft- und Raumfahrt e.V. (DLR) entstanden. Hiermit möchte ich mich bei all den Personen bedanken, die mich während dieser Zeit unterstützt haben.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Karsten Lemmer für die Übernahme der Betreuung. Die freundliche Aufnahme und die hervorragenden Bedingungen an seinem Institut haben die Entstehung dieser Arbeit erst möglich gemacht.

Herrn Prof. Dr. Ehrich möchte ich für die spontane Bereitschaft zur Teilnahme an der Prüfungskommission danken. Durch sein Engagement und Interesse war er eine prägende Autorität, die mich in wichtigen Momenten sowohl fachlich als auch moralisch unterstützt hat.

Bedanken möchte ich mich auch bei Herrn PD Dr. Frank Köster für die ständige und intensive Betreuung während der Erstellung dieser Arbeit. Die vielen konstruktiven Diskussionen und Anmerkungen haben wesentlich zum Gelingen mit beigetragen.

Weiterhin möchte ich Herrn Prof. Dr.-Ing. Dr. h.c. mult. Eckehard Schnieder für die Übernahme des Vorsitzes der Prüfungskommission danken.

Während der Umsetzung der Arbeit hatte ich ständigen Kontakt mit vielen Kollegen und Studenten, die mich sowohl fachlich als auch emotional begleitet haben und denen allen mein Dank gilt. Insbesondere hat mich Dirk Beckmann bei der Betrachtung philosophischer Fragestellungen und bei der Reflektion meiner Ergebnisse geholfen. Christoph Torens möchte ich dafür danken, dass er stets ein paar beruhigende Worte für mich übrig hatte und mir geholfen hat, die wesentlichen Dinge im Leben nicht zu vernachlässigen.

Ein ganz besonderer Dank gilt meiner Familie. Während der Bearbeitungszeit hatte ich nicht immer Möglichkeit, mich ihr in dem Maße zu widmen, wie sie es verdient. Dennoch hat sie immer zu mir gehalten und mich unterstützt, wenn es besonders notwendig war.

Ulf Noyer

Inhaltsverzeichnis

Abbildungsverzeichnis

[1] Die Abbildung wurde aus der genannten Quelle übernommen.
[2] Die Abbildung wurde in Anlehnung an die angegebene Quelle übernommen.
[3] Das Bildschirmfoto wurde selbst angefertigt und zeigt die in der Quelle angegebenen Inhalte.
[4] Zur Verbesserung der Lesbarkeit wurde der Text in der Abbildung digital nachbearbeitet.
[5] In die Abbildung wurden nachträglich Beschriftungen (in Form von gelben Rechtecken) eingefügt.
[6] Die angepasste Abbildung wurde mit freundlicher Erlaubnis des Erzeugers der Originalabbildung eingebunden.

Tabellenverzeichnis

Listing-Verzeichnis

Kurzfassung

Messdaten in Form von Zeitreihen aus wissenschaftlichen Versuchen werden zur effizienten Organisation und Weiterverarbeitung in relationalen Datenbanken gespeichert. Datenbanken bilden für diese Aufgabe eine ausgereifte und leistungsfähige Grundlage. Die in den letzten Jahren entstandenen Semantic Web Technologien bieten eine hervorragende Basis zur Wissensmodellierung und Beschreibung von Domäneninhalten in Form von Ontologien. Aufgrund der offenen Architektur dieses Ansatzes können leicht fremde Ontologien und Ressourcen mit eingebunden und berücksichtigt werden. Da Semantic Web Technologien auf Prädikatenlogik aufbauen, stellen sie eine formale Modellierungsgrundlage dar. Mittels Reasoning kann deshalb aus Ontologien implizites Wissen abgeleitet werden.

Im Rahmen dieser Arbeit werden *semantische (Datenbank-) Annotationen* und deren Interpretation fokussiert. Sie verknüpfen die Technologien relationaler Datenbanken und des Semantic Web miteinander, um deren Vorteile zu vereinen und einen zusätzlichen Mehrwert zu generieren. Diese Annotationen erlauben es, Inhalte von Datenbanken mit Semantic Web Technologien zu beschreiben. Für deren Anwendung werden verschiedene Nutzungsszenarien vorgestellt und ein Prozess entwickelt, der diese in Zusammenhang mit datenverarbeitenden Prozessen setzt. Für die gemeinsame Behandlung und den Einsatz beider Technologien wird eine Architektur entwickelt, welche insbesondere mit Hinblick auf die Skalierbarkeit beim Auftreten großer Mengen von Messdaten entworfen wird. Auf dieser Basis werden verschiedene Konzepte zur Visualisierung und Interaktion mit den semantischen Annotationen eingeführt. Weiterhin wird deren Einsatz zur formalen Modellierung von Ereignissen in Zeitreihen mittels Automaten betrachtet, sodass ein Reasoning zur Berechnung durchgeführt werden kann.

Mittels einer Implementierung werden die Umsetzbarkeit und die Vorzüge der eingeführten Konzepte demonstriert. Die vorgestellte Applikation *Semantic Database Browser* erlaubt die integrierte Verwendung von Messdaten und deren formaler Beschreibung. Formale Modelle können leicht ausgetauscht und wiederverwendet werden, sodass die Wiederverwendung von Wissen gefördert und die Zusammenarbeit effektiver gestaltet wird. Durch die Beschreibung der Messdaten mittels Modellen gewinnen sie außerdem an Verständlichkeit. Anhand des Beispiels von Ereignissen während Autofahrten wird demonstriert, wie auf Basis der formalen Beschreibung automatisch Schlussfolgerungen gezogen werden können. So können durch das Schlussfolgern ohne zusätzlichen Aufwand neue Erkenntnisse über auftretende Fahrmanöver generiert werden. Aufgrund des domänenunabhängigen Charakters der skizzierten Lösungsansätze wird gezeigt, dass diese sich leicht auf andere Anwendungsfälle anwenden lassen.

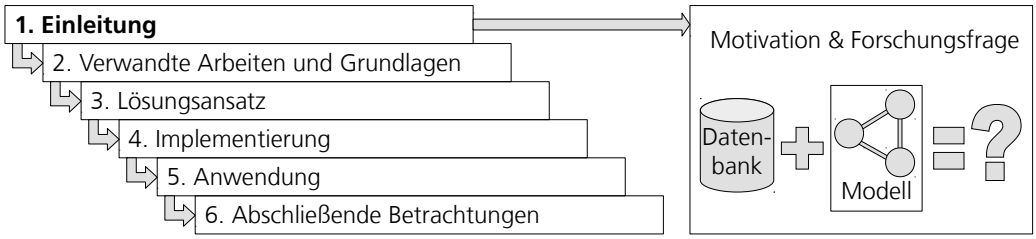
Abstract

Measurement data in form of time series of scientific experiments is stored in relational databases for efficient processing. Complementary, Semantic Web technologies have been developed in the last years for describing domain knowledge in form of ontologies. Due to their open architecture, foreign ontologies and resources can be easily referenced and integrated. Since Semantic Web technologies are based on predicate logic, they are suitable for formal modeling. Therefore, using reasoning implicit knowledge can be derived from ontologies.

This work introduces *semantic (database) annotations* to link databases and ontologies to take advantage of both together by describing database contents with Semantic Web technologies. An architecture is developed for the combined handling and usage of these two technologies, which is designed in respect of scalability of large amounts of measurement data. Based on this architecture, concepts for visualizing and interacting with annotations are introduced. Furthermore, semantic annotations are used for formally modeling events in time series using finite state machines, which are computed using reasoning.

An implementation is introduced to demonstrate the feasibility and advantages of the discussed concepts. The presented application *Semantic Database Browser* allows using semantic database annotations and interactively working with them for integrated handling of formally described measurement data. Formal models can be easily exchanged and reused to support reusability of knowledge and cooperation. By describing measurement data with models, data becomes much easier to understand. Using an example of events during driving, it is demonstrated how formal description can be used for automatic reasoning to generate additional knowledge about driving maneuvers without any additional effort. Because the presented approaches are domain independent, they can be easily adapted for other use cases.

1 Einleitung



Assistenz- und Automationssysteme tragen heute mehr und mehr zur Verbesserung der Sicherheit und des Komforts von Straßenfahrzeugen bei. Solche Systeme werden im Rahmen von umfassenden Entwicklungsprozessen erarbeitet. Bereits in frühen Phasen der Systementwicklung und im Rahmen von Systemtests greifen diese Entwicklungsprozesse z.B. auf Probandenstudien zur Erprobung der Systeme zurück, in denen umfangreiche Messdaten anfallen. Diese Arbeit beschäftigt sich mit der formalen und domänenspezifischen Beschreibung dieser Daten, die zumeist in Form von Zeitreihen vorliegen, und den in den Daten auftretenden Ereignissen.

Das Ziel dieser Beschreibung ist, die Daten und Ereignisse in Zusammenhang mit den während des Entwicklungsprozesses relevanten Konzepten und Begriffen zu setzen. Die bei diesem Prozess entstehenden Herausforderungen werden anhand des im Folgenden beschriebenen Entwicklungsprozesses erläutert.

1.1 Entwicklungsprozess von Assistenz und Automation

Der in Abb. 1-1 dargestellte Prozess wird zur Entwicklung von Assistenz und Automation ([KHFL11], weiterentwickelt aus [BCH+08]) eingesetzt. Er bildet das iterative Design von Assistenz- und Automationssystemen im Kontext verschiedener Einflussfaktoren ab. Während des Designs selber werden der Fahrer, die Umwelt, das Transportmittel und die Informations- und Kommunikationsinfrastruktur des technischen Systems berücksichtigt. So entstehen als Zwischenschritt Prototypen, welche in empirischen Versuchen getestet werden.

In dem betrachteten Entwicklungsprozess werden Prototypen und Messkampagnen auf Basis der Plattform *Dominion* [GHH08, GHHK08] umgesetzt. *Dominion* ist eine zur Verwendung in echtzeitfähigen Systemen optimierte Middleware bzw. Systemarchitektur (vgl. [SHG+10, MDD+10]). Durch *Dominion* sind Prototypen auf verschiedenen Versuchsträgern vom Simulator

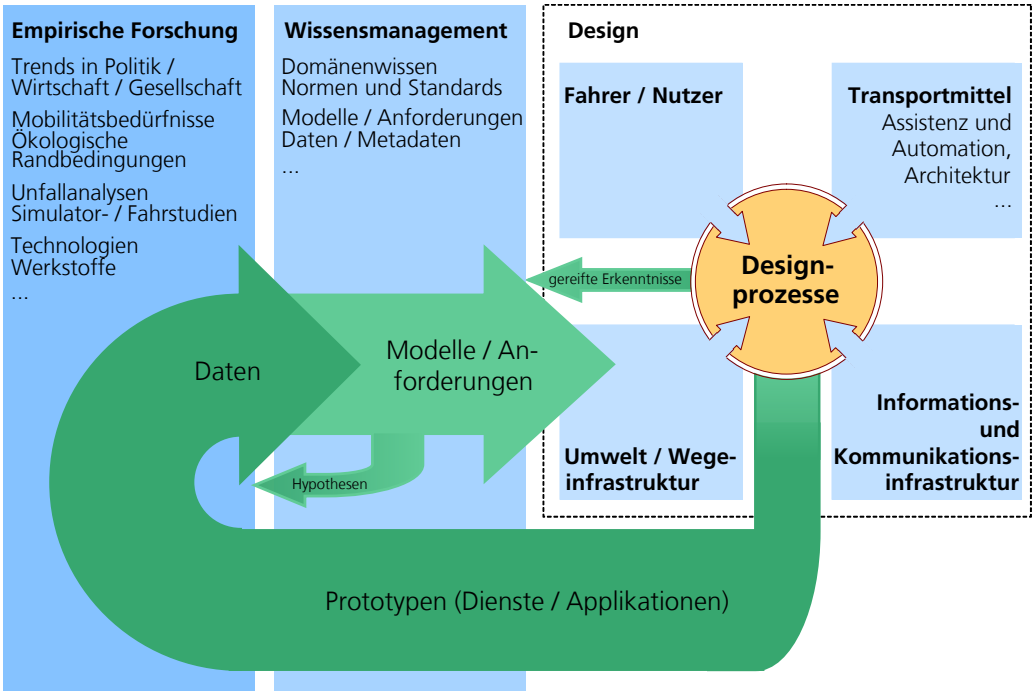


Abbildung 1-1: Entwicklungsprozess zum Design von Assistenz- und Automationssystemen[2] (nach [KHFL11])

bis zum realen Fahrzeug lauffähig. Damit können die Prototypen während der unterschiedlichen Stufen des Entwicklungsprozesses ohne zusätzlichen Anpassungsaufwand eingesetzt werden.

Weiterhin stellt *Dominion* eine einheitliche Plattform zur Aufzeichnung der Messdaten in Form von Zeitreihen dar, welche die Messdaten zur Verwaltung in einer relationalen Datenbank speichert. Relationale Datenbanken sind eine etablierte und sehr effiziente Grundlage zur Speicherung großer Datenmengen mit einer einheitlichen Struktur. Der Zugriff für Auswertungen erfolgt über standardisierte Schnittstellen, sodass gängige Werkzeuge leicht darauf zugreifen können. Die Datenbank zur Speicherung von Fahrversuchen wird *Dominion-DataStore* genannt. Die gespeicherten Messdaten müssen in Zusammenhang mit den der Entwicklung zugrundeliegenden Normen, Standards und Anforderungen gesetzt werden, welche in Form von Modellen dargestellt werden. Diese Modelle erlauben die Komplexität von Entwicklungsprozessen zu beherrschen und werden in *Ontologien* ausgedrückt.

1.2 Wissensmanagement und Ontologien

Der Begriff *Ontologie* kann zunächst als *Begriffssystem bestehend aus den Konzepten einer Wissensdomäne und deren Beziehungen untereinander* beschrieben werden (vgl. [Lz09]). Somit bilden Ontologien eine formale Beschreibung des Domänenwissens, sodass zwischen allen an der Entwicklung Beteiligten ein Konsens besteht, welche Fakten bzw. Sachzusammenhänge erörtert werden. Ontologien haben sich zur Beschreibung und Speicherung von Wissen als leistungsfähiges Werkzeug etabliert. Sie dienen ebenfalls zur Dokumentation der Daten, um

deren Nachvollziehbarkeit zu verbessern. Während des Entwicklungsprozesses werden mehrere Iterationen durchlaufen, um das Ergebnis Schritt für Schritt zu verbessern. Sind die Zusammenhänge zwischen Messdaten und dem auf Ontologien basierenden Wissensmanagement hergestellt, lassen sich die Wissensmodelle ebenfalls iterativ aktualisieren. Die Möglichkeiten von Ontologien sind hierbei sehr umfassend und vielfältig.

Wissensmanagement ist in diesem Kontext so zu verstehen, dass Wissen zu einer bestimmten Domäne verwaltet und in schriftlicher Form verfügbar gemacht wird, damit es bei Bedarf abrufbar ist (vgl.a. [AL99]). Die Speicherung und Darstellung erfolgt idealerweise mit formalen Methoden, sodass eine maschinelle Interpretation möglich ist und menschliche Anwender entlastet werden.

Die Umsetzung von Ontologien erfolgt üblicherweise mit semantischen Technologien, welche aus dem Bereich des *Semantic Web* (semantisches Netz, vgl. Abschnitt 2.3) stammen. Die Vision des Semantic Web [BLHL01, BL06a] wurde von Sir Tim Berners-Lee vorgestellt. Dieser Vision liegt die Tatsache zugrunde, dass verteilt im *World Wide Web* (WWW) ein nahezu unbegrenzter Wissensschatz existiert. Für Maschinen handelt es sich bei den im WWW verteilten Informationen jedoch um nicht wesentlich mehr als unverständlichen „Buchstabensalat". Der Zweck der sogenannten semantischen Technologien besteht darin, Maschinen den Umgang mit dem eigentlichen Informationsgehalt von Dokumenten zu ermöglichen. Auf dieser Basis sollen Computer in der Lage sein, die bereitgestellten Informationen zu kombinieren und deduktiv weitergehende Erkenntnisse aus bereits implizit vorhandenen Informationen abzuleiten. Damit wäre es möglich, den Wissensschatz im WWW automatisiert zu verwerten.

Aktuelle Arbeiten zum Semantic Web beschränken sich derzeit auf ausgewählte thematische Teilgebiete und Anwendungsdomänen. Zum Beispiel existieren in der Biologie [TAM+03], Medizin [NSW+09] oder den Verkehrswissenschaften [HYD07, BCH+08] Ansätze, die Expertenwissen formal erfassen und dabei ein vereinheitlichtes Vokabular verwenden. Diese Arbeiten legen so die Grundlagen für eine maschinell weiterverarbeitbare Wissensbasis in ihren jeweiligen Domänen.

Neben den wesentlichen Internetstandards stellt das *World Wide Web Consortium* (W3C) für das skizzierte Ziel interoperable, offene Standards für das Semantic Web bereit. Basierend auf diesen Standards lässt sich Wissen formal in Form von Ontologien beschreiben (Abschnitt 2.3). Damit wird es in Zukunft möglich sein, auf das im WWW gespeicherte Wissen zuzugreifen und gezielt Fragen zu beantworten, was eine deutliche Verbesserung gegenüber derzeitigen Internet-Suchmaschinen darstellt. Die beschriebenen Ansätze lassen aber offen, wie die Interpretation von Zeitreihen und Messdaten von diesen Ansätzen profitieren kann.

1.3 Motivation zur Beschreibung zeitstrukturierter Daten

Im Jahr 1999 wurde der weit über 100 Millionen Dollar teure Mars-Klima-Orbiter der Raumfahrtbehörde *National Aeronautics and Space Administration* (NASA) zerstört, weil Ingenieure die Maßeinheiten verwechselten. Daraufhin verglühte das Raumfahrzeug auf einem zu niedrigen Kurs. Werden numerische Werte (z.B. Messdaten) ohne ihren Zusammenhang betrachtet, können ähnliche Probleme ebenfalls bei der täglichen Arbeit mit Daten und Messwerten auftreten, auch wenn dies in der Regel nicht mit so katastrophalen Folgen geschieht. (vgl. [Nao09, SLM+99])

Anhand des Beispiels ist leicht ersichtlich, dass es essentiell ist, numerisches Material mit seiner Interpretation zu verknüpfen. Eine Verknüpfung von Messdaten und Zeitreihen mit dem Wissensmanagement und Metainformationen zu einer semantischen Datenbank erlaubt dies und ist daher erstrebenswert. Zur Modellierung dieser inhaltlichen Zusammenhänge bieten sich semantische Technologien und Ontologien geradezu an, da sie genau für diesen Zweck entwickelt wurden.

Bei semantischen Technologien handelt es sich um eine auf Graphen basierende Darstellung mit speziellem Fokus auf Offenheit und Wissensdarstellung im Internet. Somit können sie als eine spezielle Form graphbasierter Datenbanken aufgefasst werden. Relationale Datenbanken dagegen beruhen auf einer Organisation von Daten in Tabellen. Auch wenn relationale Datenbanken andere Varianten in der Vergangenheit nahezu verdrängt haben, gewinnen diese wieder an Bedeutung, da sie zur Abbildung komplexer Zusammenhänge gut geeignet sind. Folgendes ausführliche Zitat veranschaulicht dies:

> „Zu diesem Revival [der Graphdatenbanken] trägt die Veränderung des Internet von einem Dokumentensilo zu einem globalen Graphen bei, der Inhalte verbindet. Laut Tim Berners-Lee setzt dieser Graph Informationen aus sozialen Netzen, Blogs und Orten in Beziehung und verwendet sie auf von den Autoren nicht vorgesehene Weise [BL07]. Daraus ergeben sich neue Anforderungen an die Verarbeitung dieser vernetzten Daten. Der Fokus verschiebt sich von den Daten (Knoten) zu ihren Beziehungen (Kanten).
>
> Das klassische, relationale Datenbankmodell vermag einfache Datensätze nach einem festen Schema abzulegen und effizient zu verwalten. Für komplexere Anfragen, etwa „Welche Artikel haben Freunde meiner Freunde gestern gekauft?“ eignet sich das Modell nicht oder schlecht [AG08]. Bei steigender Komplexität der Daten und Abfragen, zum Beispiel Abbildungen von Baumstrukturen, Hierarchien und Netzen, lässt die Leistung relationaler Datenbanken schon bei Webanwendungen und sozialen Netzen nach [Eif10]. Im Produktivbetrieb ist das statische Datenmodell kaum modifizierbar, und Änderungen beeinflussen die darüberliegende Anwendung meist direkt. [. . .]

> [Ein graphbasiertes Daten-] Modell lässt sich zwar in das relationale überführen. Allerdings ist das bei Strukturen wie Bäumen und Graph-Relationen schwierig [. . .]. Dies verdeutlicht das triviale Beispiel *Freund eines Freundes*. Solche Beziehungsgraphen lassen sich in zwei Tabellen darstellen: Eine enthält die personenbezogenen Daten, die andere die Beziehungen zwischen den Personen. Jede Beziehungskante repräsentiert mindestens eine [..] Bedingung, die eine Mengenoperation auf den Tupeln dieser beiden Tabellen erfordert. Anfragen nach *Freund eines Freundes*-Beziehungen führen zu nicht skalierenden Joins [in Form von sehr komplexen Abfrageausdrücken]. Deshalb sind Tabellen für stark verknüpfte Daten wenig geeignet. Ihre Verarbeitung ist die Stärke der Graph-Datenbanken [z.B. in Form von semantischen Technologien]."[Rit10, S. 78 f.]

Aufgrund der einfachen und tabellarischen Strukturierung von Messdaten sind relationale Datenbanken zur Verwaltung von Messdaten ideal geeignet. Dies gilt allgemein für Massendaten, welche eine einfache Struktur haben und in großen Mengen auftreten. Selbst für große Tabellen sind relationale Datenbanken effizient. Weiterhin lassen sie sich über Standardschnittstellen anbinden, was von verbreiteten Softwarewerkzeugen unterstützt wird und für einen operativen Betrieb unverzichtbar ist. Aus diesen Gründen ist es nicht sinnvoll, auf eine relationale Datenbank für das angestrebte Anwendungsszenario zu verzichten.

Wie im obigen Zitat ausgeführt, eignen sich semantische Technologien dagegen besonders zur offenen und modularen Modellierung komplexer Beziehungen, wie sie u.a. beim Wissensmanagement auftreten. Das Internet besteht aus vielen separaten Wissensinseln, welche als Webseiten von verschiedenen Urhebern angeboten werden. Als Schöpfung des Internets orientieren sich semantische Technologien an den Technologien des Internets. Somit können mittels semantischer Technologien und Ontologien beschriebene Wissensbasen modularisiert werden. Da sie auf entsprechenden Designprinzipien beruhen (Namensräume, graphenbasiertes Datenmodell etc.), können beliebige Wissensbasen miteinander kombiniert werden (Abschnitt 2.3). Hierdurch ergibt sich die Möglichkeit, fremde Elemente weiterzuverwenden oder eigene Elemente zur Weiterverwendung zu veröffentlichen. So können Vokabulare abgeglichen und eine Interoperabilität mit anderen Datenbeständen ermöglicht werden. Auch lassen sich Ausschnitte mittels semantischer Technologien beschriebener Zusammenhänge direkt als visueller Graph oder als Topic Map visualisieren (vgl. [GS02b]). Weiterhin lassen sich aufgrund der formalen Darstellung modellierten Wissens *Reasoner* einsetzen, um implizites Wissen automatisiert abzuleiten (Abschnitt 2.3.6.2). Somit sind semantische Technologien ideal geeignet, (abstrakte) Wissenszusammenhänge in Bezug auf Massendaten und Messdaten zu beschreiben.

Die unterschiedlichen Stärken von relationalen Datenbanken und semantischen Technologien sind jeweils zusammenfassend in Tab. 1-1 aufgeführt. Eine Kombination einer relationalen Datenbank zur Speicherung von großen Datenmengen mit semantischen Technologien zur formalen Beschreibung erlaubt es, die unterschiedlichen Stärken dieser Technologien zu kombinieren.

Relationale Datenbanken	Semantische Technologien
Domänenunabhängig	Domänenunabhängig
Besonders leicht anwendbar bei tabellenähnlicher Struktur	Direkte Abbildung komplexer Strukturen (z.B. Hierarchien)
Visualisierung von Daten als Tabellen	Visualisierung von Beziehungen als Graphen
Fokus auf effiziente Behandlung großer Datenmengen	Fokus auf offenen und modularen Ansatz zur Verknüpfung vieler Datenquellen
Anbindung über standardisierte und etablierte Schnittstellen	Vorhandene Schnittstellen noch nicht etabliert
Implizites Wissen nicht direkt zugreifbar	Automatisierte Ableitung impliziten Wissens (Reasoning)

Tabelle 1-1: Stärken von relationalen Datenbanken und semantischen Technologien

Durch diese Verknüpfung profitiert somit das Datenmanagement von Messdaten deutlich. Aus dieser Herausforderung leiten sich die Zielsetzung und die Forschungsfrage dieser Arbeit ab.

1.4 Zielsetzung, Forschungsfrage und Beitrag der Arbeit

Es gilt, die Vorteile von relationalen Datenbanken und semantischen Technologien durch eine geeignete Kombination für die Verwaltung von Massendaten und darauf bezogene Zusammenhänge nutzbar zu machen. Abbildung 1-2 skizziert, wie das Zusammenspiel der zugrundeliegenden Technologien in einer gemeinsamen Anwendung aussehen könnte. Da beide zugrunde liegenden Technologien domänenunabhängig sind, sollte dies idealerweise ebenfalls für das Ergebnis zutreffen. Eine Ausgestaltung auf Konzeptions- und Architekturebene mit dem Ziel der gemeinsamen Verwendung spiegelt sich in der folgenden Fragestellung wider.

Das Ziel dieser Arbeit besteht in der formalen und domänenspezifischen Beschreibung von Messdaten in einer relationalen Datenbank, um deren Interpretation und Wiederverwendbarkeit zu unterstützen.

Die zentrale Fragestellung, welche dieser Arbeit zugrunde liegt, ergibt sich wie folgt:

> Wie können in einer Datenbank gespeicherte numerische Daten (z.B. Zeitreihen) mit Domänenwissen, zur Wahrung der ursprünglichen und abgeleiteten Interpretation, skalierbar verknüpft werden und dem Anwender bedienfreundlich präsentiert werden, sodass diese Kontextinformationen in formaler Form bei der Arbeit mit den Daten verfügbar sind?

Aus dieser übergeordneten Frage ergeben sich mehrere detailliertere Fragen zu unterschiedlichen Bereichen:

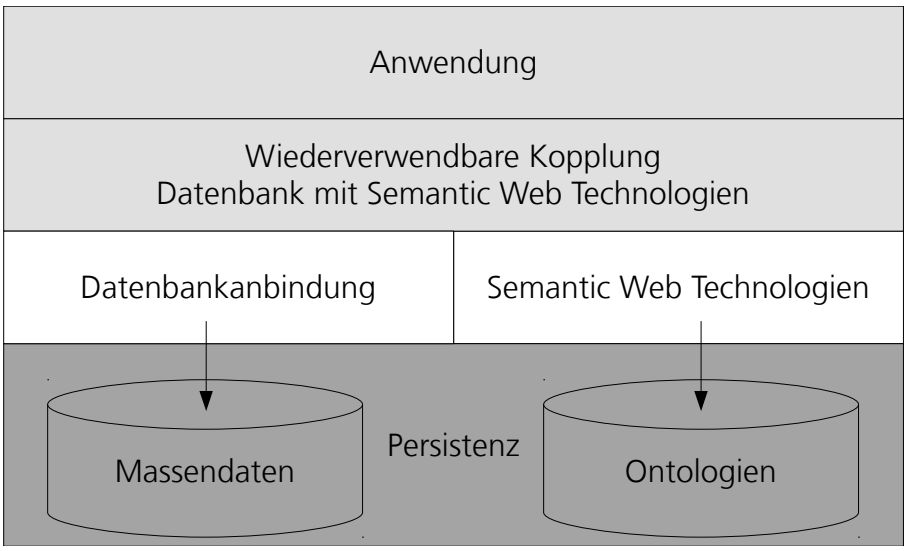


Abbildung 1-2: Architekturskizze zur Kopplung von Massendaten mit Ontologien

Konzept Wie lassen sich Datenbankinhalte und semantische Technologien konzeptionell miteinander verknüpfen? Wie können klassische numerische Daten mit semantischen Annotationen in einem gemeinsamen Prozess behandelt werden?

Architektur und Visualisierung Wie muss eine Architektur zur gemeinsamen Datenhaltung aussehen, damit sie bei großen Datenmengen skaliert? Wie kann die Einbettung der semantischen Zeitreihenbeschreibung zur gemeinsamen Benutzung in Anwendungen in einem verteilten System aussehen? Wie lässt sich eine semantische Beschreibung in Form von Annotationen so visualisieren, dass die Datenbestände interaktiv exploriert werden können?

Anwendung Wie können Zeitreihen so mit semantischen Technologien beschrieben werden, dass die auftretenden Ereignisse über Reasoning miteinander in Beziehung gesetzt werden können? Wie eng ist bei den eingeführten Konzepten und Methoden die Kopplung an eine bestimmte Domäne?

Der wissenschaftliche Beitrag der vorliegenden Arbeit liegt in der Behandlung dieser Menge eng miteinander verknüpfter Fragen. Bei der Diskussion werden verschiedene Ebenen von der Architektur über die Behandlung von Projektionen zur Visualisierung bis hin zur Wissensmodellierung behandelt.

Anhand einer prototypischen Implementierung wird die Umsetzbarkeit der erarbeiteten Konzepte demonstriert. Insbesondere die vorgeschlagenen Konzepte zur Nutzerinteraktion lassen sich anhand der Implementierung erproben. Beispiele und Anwendungsfälle für die betrachteten Zeitreihen werden aus der Domäne von Fahrversuchen zur Entwicklung von Assistenz- und Automationssystemen gewählt. Der Fokus liegt hierbei auf der Beschreibung von Fahrmanövern.

1.5 Aufbau der Arbeit

Der Aufbau dieser Arbeit ist in Abb. 1-3 dargestellt. Zu Beginn eines jeden Kapitels ist eine an diese Abbildung angelehnte Grafik zu finden, um die Orientierung zu erleichtern. Im folgenden Kapitel 2 werden zunächst Arbeiten mit einem thematischen Bezug zu dieser Arbeit diskutiert und die benötigten Grundlagen der eingesetzten Technologien behandelt. Aufbauend auf den zuvor eingeführten Grundlagen wird in Kapitel 3 ein Lösungsansatz vorgestellt, der die in Abschnitt 1.4 gestellte Forschungsfrage zum gemeinsamen Einsatz von Datenbanken und semantischen Technologien bearbeitet. Es wird ein Konzept erstellt, wie diese Technologien vorteilhaft kombiniert werden können, um das Datenmanagement von Messdaten zu verbessern. Semantische Technologien werden zur formalen Beschreibung der Datenbestände eingesetzt und in einen Prozess integriert. Weiterhin wird eine Architektur entwickelt, um die gemeinsame Behandlung von Messdaten und deren Beschreibung skalierbar zu gestalten.

Eine Implementierung, die die vorgestellten theoretischen Ansätze zur Demonstration in ein Werkzeug umsetzt, wird in Kapitel 4 beschrieben. Kapitel 5 behandelt die praktische Anwendung der erarbeiten Konzepte mit Hilfe des entstandenen Werkzeuges auf die Domäne von Fahrversuchen. Abschließend wird in Kapitel 6 eine Zusammenfassung und Bewertung der durchgeführten Betrachtungen und ein Ausblick über mögliche weitergehende Anschlussfragestellungen gegeben. Ergänzende Angaben, auf welche im Text hingewiesen wird, befinden sich im Anhang.

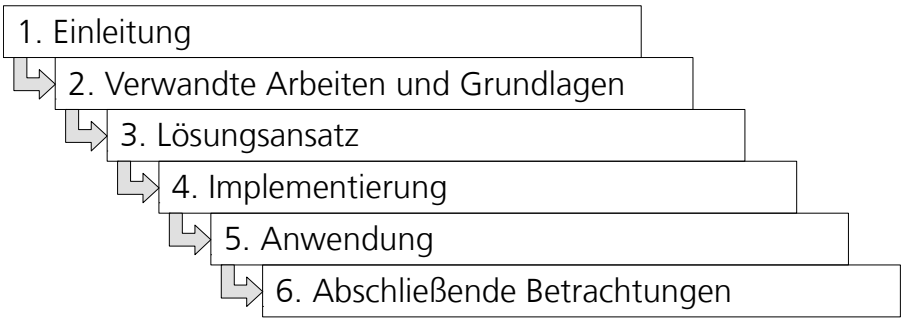
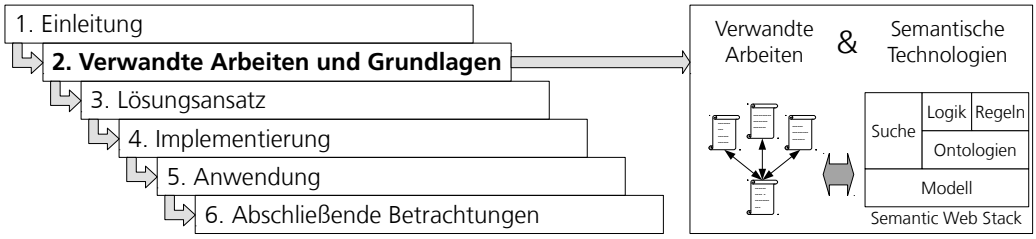


Abbildung 1-3: Struktureller Aufbau der Arbeit

2 Verwandte Arbeiten und Grundlagen



In diesem Kapitel werden Themen erörtert, die mit dem diskutierten Thema zur gemeinsamen Behandlung von Messdaten und formalen Modellen verwandt oder zum weiteren Verständnis dieser Arbeit notwendig sind. Zunächst werden im Abschnitt 2.1 grundlegende Annahmen und Voraussetzungen geklärt. Im Anschluss daran wird in Abschnitt 2.2 der Bezug anderer Arbeiten zur vorliegenden Aufgabenstellung herausgearbeitet und bewertet, wobei ausgewählte Arbeiten und Aspekte fokussiert werden. In Abschnitt 2.3 werden dann Grundlagen und Möglichkeiten semantischer Technologien vorgestellt.

2.1 Annahmen und Voraussetzungen

Diese Arbeit behandelt das Datenmanagement für Messdaten, wobei die Anwendungsfälle aus dem Bereich von Versuchsfahrten [KN08] gewählt sind. Dabei werden zum einen der Inhalt und die Interpretation von Versuchsfahrten betrachtet. Dazu gehören z.B. Fahrmanöver [Nag94], Zustandsvariablen oder der Zustand des Fahrers [VSA+05, Sch09a]. Andererseits wird die Beschreibung der Versuche zum Zwecke einer langfristig verfügbaren und aussagekräftigen Dokumentation behandelt. Deren Bedeutung ist insbesondere für größer dimensionierte Entwicklungsprozesse [GJBK08] als auch für länger andauernde Studien hervorzuheben (Naturalistic Driving Studies, Long Term Studies oder Field Operational Tests, [NSU+13, FES+98, ESL+05b, ESL+05a]).

Als Grundlage zur Speicherung der Messdaten wird in dieser Arbeit eine *Datenbank* eingesetzt. Eine Datenbank ist eine logisch zusammenhängende Datenbasis und stellt die eigentlichen Nutzdaten dar. Ein *Datenbankmanagementsystem* (DBMS, [KE06, Kap. 1, S. 17 ff.]) verwaltet dabei eine oder mehrere Datenbanken. In dieser Arbeit wird ein *relationales Datenbankmanagementsystem* (RDBMS) verwendet. Relationale Datenbanken bilden eine solide Grundlage zur Datenspeicherung, sind weit verbreitet und können als der derzeitige Stand der Technik für

diese Aufgabe betrachtet werden. Das grundlegende Konzept zur Datenorganisation in relationalen Datenbanken ist das relationale Modell. In diesem Modell werden in den Relationen, welche Tabellen entsprechen, die Daten gespeichert. Jede Zeile ist ein Datenobjekt bzw. Datensatz. Die Eigenschaften bzw. Spalten einer Tabelle heißen Attribute.

Darüber hinaus gibt es noch andere Datenbankkonzepte ([KE06, Kap. 1.6.2, S. 23 ff.], z.B. [Heu97, Gor06]), die nicht relational sind. Diese haben sich aber nicht im gleichen Maße durchgesetzt oder sind derzeitig Gegenstand der Forschung.

Jedoch besitzen relationale Datenbanken häufig Objekterweiterungen, mit denen eigene Datentypen und Operationen auf diesen definiert werden können. Aus diesem Grund wird bei relationalen Datenbanken mit entsprechenden Erweiterungen von objekt-relationalen Datenbanken [KE06, Kap. 14, S. 397 ff.] gesprochen. Aufgrund der stark überwiegenden Verbreitung ist im weiteren Verlauf bei der Verwendung des Ausdrucks Datenbank — soweit nicht anders angegeben — eine Datenbank in einem relationalen Datenbankmanagementsystem gemeint.

Datenbanken besitzen verschiedene Vorteile gegenüber proprietären Speicherformaten oder einfachen dateibasierten Formaten, wie z.B. Textdateien oder durch Kommas getrennte Dateien (Comma-separated Values, CSV). Zum einen stellt das DBMS sicher, dass die tabellarische Struktur der Daten niemals verletzt wird. Ein anderer wesentlicher Vorteil ist die Möglichkeit des Zugriffs über die standardisierte Abfragesprache *Structured Query Language* (SQL, [CB74, Int92, OGC99]). Über vereinheitlichte Programmierschnittstellen (z.B. Java Database Connectivity (JDBC) [KS08a, Kap. 42, S. 987 ff.], Open Database Connectivity (ODBC) [Gei99]) lässt sich aus beliebigen Anwendungen mittels SQL auf Datenbanken verschiedener Hersteller zugreifen.

Darüber hinaus bieten Datenbanken noch zusätzliche Konzepte, um die Integrität der verwalteten Daten sicherzustellen und die Arbeit mit ihnen zu vereinfachen (Transaktionen, Schlüssel, Constraints, Trigger etc. [KE06]). Auch bieten sie mit Indizes und ausgefeilten Abfrageoptimierern leistungsfähige Mechanismen, um den Zugriff auf eine transparente Art und Weise deutlich zu beschleunigen. Da Datenbanken eine ausführliche Auskunft über die von ihnen verwalteten Inhalte geben, lassen sich weiterhin Anwendungen entwickeln, die sich flexibel an die Tabellenstruktur einer Datenbank anpassen.

Verfügbare Datenbanken sind somit ausgereifte Produkte, die große Datenmengen effizient handhaben können (vgl.a. [NRR07, PA04]) und sich nicht einfach durch eine selbst implementierte Lösung sinnvoll ersetzen lassen. Aus diesem Grund bilden Datenbanken für die Betrachtungen zum Datenmanagement die Grundlage.

2.2 Verwandte Arbeiten

Dieser Abschnitt setzt die vorliegende Arbeit in einen Zusammenhang mit anderen Arbeiten, welche in Bezug zur behandelten Fragestellung zur Beschreibung von Messdaten stehen. Für die in der Einleitung genannten Beispiele von Fahrversuchen oder dem zerstörten Mars-Klima-Orbiter würden hier somit Arbeiten betrachtet werden, welche dort anfallende Messdaten in Verbindung mit beschreibenden Metadaten setzen. Das Ziel ist die Vermeidung von Datenfehlinterpretation, welche zum Beispiel zum Absturz eines Raumfahrzeuges führen können.

Zunächst wird auf die allgemeine Bedeutung von Metadaten eingegangen, da sämtliche Bereiche der verwandten Arbeiten diesen Themenkomplex berühren. Anschließend werden diese Bereiche miteinander verglichen, welche unterschiedliche Schwerpunkte beim Datenmanagement setzen. So können sie in Bezug zu den in dieser Arbeit behandelten Inhalten gesetzt werden und es kann von ihren verschiedenen Vorgehensweisen und Erkenntnissen profitiert werden.

Im Anschluss an die Metadatendiskussion werden die verschiedenen Einsatzszenarien von Datenbanken (Abschnitt 2.2.2), *Scientific Workflow Systems* (SWS, Abschnitt 2.2.3), wissenschaftlichen Datenhaltungssystemen (Abschnitt 2.2.4) und aktuellen Ansätzen zur semantischen Annotation (Abschnitt 2.2.5) miteinander verglichen. Anhand der betrachteten Einsatzszenarien erfolgt jeweils eine tabellarische Charakterisierung der genannten Bereiche.

Diese Bereiche werden nach der Leistungsfähigkeit der *Datenverwaltung* für die Massendaten und der Möglichkeit zur *Interpretation der Dateninhalte* verglichen. Ein weiteres Kriterium ist die Ausdrucksstärke und Erweiterbarkeit des *Datenschemas*. Bezüglich der *Metadaten* werden deren Interoperabilität und Ausdrucksvermögen beurteilt. Sowohl für die Massendaten als auch für die Metadaten ist die *Durchsuchbarkeit* und *Betrachtung der Datenentstehung* von Interesse. Die spezielle *Behandlung ortsbezogener Daten* ist nötig, wenn deren zweidimensionaler Charakter und die Darstellung auf Karten berücksichtigt werden sollen. In der *Benutzerschnittstelle* ist die informative und ansprechende Aufbereitung der betrachteten Daten relevant. Abschließend wird die *Domänenunabhängigkeit* der verschiedenen Ansätze zum Datenmanagement verglichen und bewertet.

2.2.1 Rolle von Metadaten

Der Begriff Metadaten wird allgemein als *Daten über Daten* charakterisiert (vgl. [Kös07, Kap. 2, S. 30 ff.]), wobei je nach Einsatzbereich spezialisiertere Definitionen zu finden sind. In diesem Abschnitt wird in Anlehnung an [Rot96] betrachtet, wie geeignete Metadaten die Datenqualität und Langlebigkeit eines Datenproduktes unterstützen. Viele Metadaten dieser Kategorie müssen von den Datenproduzenten zur Verfügung gestellt werden. Sie beschreiben das gelieferte Datenprodukt und erlauben so, dessen Qualität zu evaluieren und zu verbessern. Eine

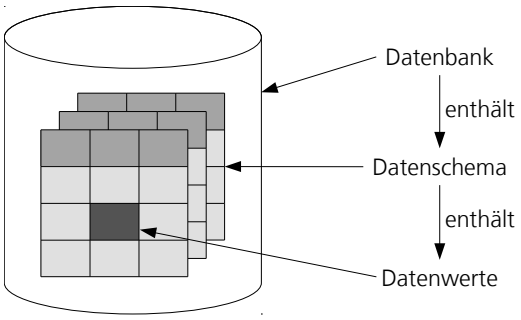


Abbildung 2-1: Abstraktionsebenen von Datenprodukten für den Einsatz von Metadaten

Strategie ist die explizite Kontrolle der produzierten Daten. Als paralleler Ansatz können Metadaten einen organisatorischen Prozess zur Datenerhebung und -verarbeitung steuern oder die Durchführung und die Ergebnisse dokumentieren.

Die verschiedenen Abstraktionsebenen zur Beschreibung von Datenprodukten durch Metadaten zeigt Abb. 2-1. Hierbei umfasst die betrachtete Datenbank die enthaltenen Datenstrukturen, welche als Datenschema bezeichnet werden. Diese wiederum enthalten die Datenwerte. Eine mögliche Systematisierung für den Verwendungszweck von Metadaten in Bezug auf Datenprodukte ist im Folgenden in Anlehnung an [Rot96, Kap. 5] aufgeführt:

- Datenbank-Ebene
  - Allgemeine Informationen zur Datenbank: Beschreibung, Verwendungszweck, Verwendungsvoraussetzungen, Datenbankstruktur
  - Quelleninformationen: Quelle(n) und deren Verlässlichkeit, Verantwortlichkeit
  - Charakterisierung: Angestrebter Detaillierungsgrad und Qualität
  - Datenabhängigkeiten: Abhängigkeiten von Datenelementen, Wertebereichseinschränkungen, statistische Verteilung der Datenwerte
  - Gemessene Qualität: Korrektheit, Vollständigkeit, Konsistenz, Aktualität, Redundanz, Einheitlichkeit, Genauigkeit
  - Prozesssteuerung: Konfiguration und Parametrisierung

- Datenschema-Ebene
  - Bedeutung: Entsprechung in der Realität, Bedeutung von NULL (kein Wert existent)
  - Quellen- und Aktualisierungsinformationen: Erlaubt Kombination mehrerer Quellen
  - Transformationen: Aggregationen, Transformationsprozess
  - Abhängigkeiten zu anderen Datenbanken: Vollständigkeit übernommener Attribute
  - Domäne und Datentyp: Maßeinheit, Restriktionen in Verwendung
  - Granularität: Auflösung, Präzision, angestrebte Genauigkeit
  - Änderungshistorie: Evolution Domäne/Typ/Einheiten, Modifikationen
  - Auditierungsergebnisse: Verwendbarkeit, Bewertung

- Datenwert-Ebene
  - Gemessene Qualität: Korrektheit, Vollständigkeit, Konsistenz, Aktualität, Redundanz, Einheitlichkeit, Genauigkeit
  - Annotationen: Widersprüche, Inkonsistenzen, spezielle Werte
  - Quelleninformationen: Erfassungszeitpunkt, Quelle, Abstammung
  - Nächste Quelle: Erwartete Aktualisierungen
  - Transformationen: Aggregationen, Modifikationen, Verarbeitungsschritte, Prozessparameter
  - Transformationshistorie: Alte Werte, Änderungszeitpunkte, derzeit laufende Transformationen
  - Auditierungsergebnisse: Durchgeführte Audits, Betrachtungsgegenstand

Die Langlebigkeit eines Datenproduktes wird durch die Lebenszeit des Speichermediums und der Soft- und Hardware zur Interpretation der Daten beschränkt. Durch regelmäßiges Duplizieren der Daten auf neue Datenträger kann theoretisch beliebig lange die physikalische Existenz der Daten sichergestellt werden. Jedoch sind viele Daten nach einiger Zeit nicht mehr verarbeitbar, weil entweder aktuelle Software sie nicht mehr verarbeiten kann oder die Bedeutung der Datenwerte verloren gegangen ist. Dieser Umstand hat zu der ironischen Aussage geführt, „that digital information lasts forever — or five years, whichever comes first“ [Rot96, Kap. 1]. Gerade hier können Metadaten einen wesentlichen Beitrag leisten, um die Bedeutung der Elemente des Datenproduktes (formal) zu beschreiben und dessen Interpretierbarkeit langfristig sicherzustellen.

Ausschließlich mit der Nachvollziehbarkeit durchgeführter Transformationen auf einem Datenprodukt beschäftigt sich die Teildisziplin des *Data Provenance* bzw. *Data Lineage* (vgl.a. [WS97, CW00, CW01, BKT01, CJ07]). Hierbei werden die auf den Daten durchgeführten Operationen in einem Graphen dargestellt, um sie besser verwalten und verarbeiten zu können. Einige Arbeiten beschäftigen sich mit den speziellen Anforderungen an Metadaten für statistische Daten, um deren Verwendung in einem sachlich richtigen Kontext zu garantieren, z.B. [GAWPL96, GA99, Nat04].

Letztlich handelt es sich bei Metadaten um Daten, sodass diese nicht immer eindeutig auseinander gehalten werden können. Das hat zur Folge, dass viele Überlegungen, die für Daten angestellt werden, auch auf Metadaten zutreffen. Beispielsweise ist der Entwurf eines Modells für Metadaten ebenso anspruchsvoll wie für normale Daten (vgl.a. [SR96, PH04]). Auch muss natürlich die Qualität von Metadaten und nicht nur die Qualität der Daten sichergestellt werden.

2.2.1.1 Bewertung der Rolle von Metadaten

In diesem Abschnitt wird generisch aufgeführt, dass Metadaten das Datenmanagement in vielen Aspekten entscheidend unterstützen, z.B. der Dokumentation, der Interpretation, Archivie-

rung und Nachvollziehbarkeit. Anhand verschiedener Ebenen werden systematisch deren konkrete Einsatzmöglichkeiten genannt.

Im Rahmen dieser Arbeit werden Semantic Web Technologien zur Darstellung von Metadaten für Messdaten eingeführt, sodass die skizzierten Anwendungsfälle ebenfalls für Messdaten umgesetzt werden können. Somit ergeben sich die genannten vielfältigen neuen Möglichkeiten zur Verbesserung des Datenmanagements. Die nächsten Abschnitte betrachten zunächst, wie derzeit andere Vorgehensweisen und Werkzeuge das Datenmanagement, Metadaten und die Datenverarbeitung in den genannten Aspekten unterstützen. Dort wird gezeigt, welche zusätzlichen Chancen sich durch den Einsatz von Metadaten und Semantic Web Technologien in diesem Bereich bieten.

2.2.2 Einsatz von Datenbanken zur Unterstützung der Datenanalyse

Datenbanken dienen als modernes Konzept zur strukturierten Speicherung großer Datenmengen. Sie bilden die Grundlage für darauf aufbauende komplexe Anwendungen und Informationssysteme (vgl.a. [SR13, BG09]). Ein Informationssystem ist ein rechnergestütztes System mit einer Mensch-Maschine-Schnittstelle, welches der Erfassung, Speicherung, Verarbeitung und Anzeige von Informationen bzw. Daten dient (vgl. [SR13, S. 4 u. 10]). Der Einsatz einer Datenbank für einen solchen Anwendungsfall bedarf einer sorgfältigen Planung des Datenmodells, welches die Datenbank verwaltet und durch die Anwendung verarbeitet wird. So kann eine Datenbank die Basis zur Speicherung in einem SWS darstellen oder aber bei dem Fall der semantischen Indizierung von Daten eine Rolle spielen.

Im Laufe der Zeit haben sich spezielle Disziplinen zum Einsatz von Datenbanken herausentwickelt. Ursprünglich aus betriebswirtschaftlichen Anwendungsszenarien stammt das Konzept der *Data-Warehouses* (DWH, [BG09, CD97, Koc08]). William Harvey Inmon, der als Vater des Data-Warehousing genannt wird, definiert es in [Inm02, Kap. 2, S. 31] wie folgt: „A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions." Das Ziel von DWH ist, Daten aus verschiedenen Datenquellen zu integrieren, um einen umfassenden Blick zu einem bestimmten Thema zu erhalten. Außerdem steht eine langfristige Speicherung der Daten im Fokus, um zeitliche Entwicklungen interpretieren und nach Möglichkeit Projektionen für die Zukunft durchführen zu können. Insbesondere unter betriebswirtschaftlichen Fragestellungen möchte man Entscheidungshilfen zur Unternehmensführung finden. In diesem speziellen Zusammenhang wird von *Business Intelligence* gesprochen [KMU06, S. 4]: „Unter Business Intelligence [. . .] werden alle direkt und indirekt für die Entscheidungsunterstützung eingesetzten Anwendungen verstanden. Dieses beinhaltet neben der Auswertungs- und Präsentationsfunktionalität auch die Datenaufbereitung und -speicherung."

Die Gemeinsamkeiten dieser Anwendungsgattung und dem vorliegenden Anwendungsszenario bestehen insbesondere darin, dass große Datenmengen über einen langen Zeitraum ein-

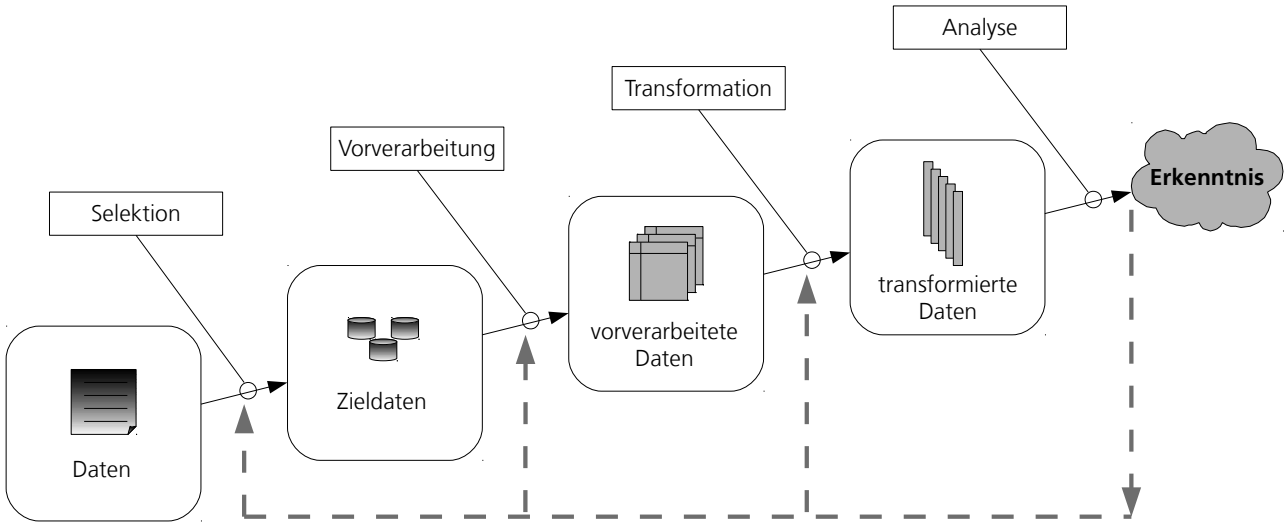


Abbildung 2-2: Schritte der Datenaufbereitung als Vorbereitung für weitergehende Analysen[2] (basierend auf [FPSS96])

bezogen werden müssen. Zur Behandlung der auftretenden Herausforderungen haben sich einige Referenzarchitekturen etabliert, die z.B. in [BG09, Kap. 2, S. 33 ff.] diskutiert werden. Ein separat behandelter Aspekt ist die Integration der relevanten Daten als Vorbereitung für weitergehende Analysen in einem iterativen Prozess (vgl.a. [CGL+99]). Abbildung 2-2 zeigt die relevanten Schritte zur Datenaufbereitung in Anlehnung an [FPSS96] und [Kös07, Kap. 3.3, S. 47 ff.]. Die linke Seite der Abbildung zeigt die in der Literatur diskutierten Schritte zur Datenaufbereitung in Form von Selektion, Vorverarbeitung und Transformation, an die sich dann die Datenanalyse anschließt.

Ein weiterer Themenkomplex aus dem Data-Warehouse-Bereich beschäftigt sich mit der Beschreibung des Aufbaus von Data-Warehouses durch Metadaten (vgl.a. [SVV99, SMR99]). Dieser hat sich schnell als so bedeutsam erwiesen, dass es sehr bald erste Bestrebungen in Richtung Standardisierung gegeben hat (vgl.a. [Obj01, AvMH02]). Ergebnis ist das *Common Warehouse Metamodel* (CWM, [Obj03]), das erstmals 2001 verabschiedet wurde und seit 2003 in der derzeitig aktuellen Version 1.1 vorliegt. Besonders erwähnenswert ist, dass CWM auf Basis des *XML Metadata Interchange* (XMI, [Obj05b]) Formats spezifiziert ist, welches auf der *Extensible Markup Language* (XML) basiert. XMI beruht auf einer Meta-Meta-Ebene [Obj05a, Kap. 6.2], auf deren Basis sich Metamodelle (z.B. CWM) spezifizieren lassen. Ein anderes sehr bekanntes Beispiel einer Metasprache auf Basis von XMI ist die *Unified Modeling Language* (UML, [Obj12a, Obj12b, RQZ07]). Dieser Ansatz ermöglicht es, Softwarekomponenten zur gemeinsamen Verwendung für XMI-Metamodelle zu entwickeln. Mittels CWM lassen sich Aufbau, Prozesse und Aggregationssichten eines Data-Warehouse beschreiben. Für die einzelnen Aspekte gibt es entsprechende Teilspezifikationen. Phänomene innerhalb der Nutzdaten auf Ebene der Datenwerte können jedoch mittels CWM nicht beschrieben werden, weswegen im weiteren Verlauf nach einer leistungsfähigeren Lösung zur Modellierung der Metadaten gesucht werden muss.

In der Literatur ist weiterhin eine separate Betrachtung für die Verarbeitung wissenschaftlicher Daten in DWH-Systemen zu finden. Eine Herausforderung können je nach Anwendungsfall die sehr großen Datenmengen sein (vgl.a. [CCD+04]). Weitere besondere Beachtung finden in diesem Zusammenhang i.d.R. spatio-temporale (räumliche und zeitliche) Aspekte (vgl.a. [MZ08]). Die Zeit verdient diese besondere Beachtung, da Phänomene bei Sensordaten nicht nur als singuläre Ereignisse auftreten, sondern sich insbesondere in Sequenzen und Abläufen niederschlagen. Bezüglich räumlicher Aspekte werden in der Regel geometrische Figuren und deren Relationen zueinander betrachtet, was komplexer ist als die Betrachtung skalarer Variablen. Aus diesem Grund existieren Forschungsansätze für den Entwurf spezieller Anfragesprachen, um spatio-temporale Phänomene in den Nutzdaten zu beschreiben (z.B. [CZ00, ES02, OGC99]). Daher lassen sich die standardisierten Verdichtungsoperationen aus betriebswirtschaftlichen DWH in vielen Fällen nicht übertragen. Somit handelt es sich bei der Interpretation solcher Daten selber um einen wissenschaftlichen Prozess und der Schwerpunkt bei dem Entwurf dieser Systeme liegt mit einem stärkeren Fokus auf der eigentlichen Datenhaltung. Insbesondere wenn sich im Laufe der Zeit die Struktur der zugrundeliegenden Daten stark ändert, ist dies eine zusätzliche Herausforderung.

Ein weiteres Thema, welches oft in Zusammenhang mit DWH-Systemen genannt wird, ist das *Data-Mining* [CCK+00] und das *Knowledge Discovery in Databases* [Kös07, Kap. 3, S. 39 ff.]. Dabei handelt es sich um einen Oberbegriff, unter dem systematische Methoden zusammengefasst werden, die das Ziel haben, Muster in Datenbeständen zu erkennen und so bisher unbekannte Zusammenhänge zu entdecken. Diese Verfahren beruhen auf statistisch-mathematischen Zusammenhängen und bedürfen daher möglichst großer zu betrachtender Datenmengen für eine optimale Anwendung. Somit bilden DWH-Systeme natürlich eine gute Grundlage für Data-Mining-Anwendungen. Bei der Betrachtung von Prozess- und Sensordaten sind insbesondere zeitliche Muster von Interesse, sodass spezialisierte Verfahren benötigt werden (vgl.a. [SP05, EA12]). Das Thema Data-Mining in Bezug zu Metadaten geht jedoch über den Gegenstand der vorliegenden Arbeit hinaus, kann aber Gegenstand zukünftiger Betrachtungen sein.

2.2.2.1 Datenbankzentrierte Informationssysteme für Fahrversuche

Nachdem der letzte Abschnitt allgemein in das Thema datenbankbasierte Informationssysteme eingeführt hat, folgen jetzt einige Beispiele aus dem Bereich der betrachteten Anwendungsdomäne für die Fahrdatenanalyse. Diese Entwicklungen sind natürlich sehr speziell und nicht unbedingt auf andere Domänen übertragbar, jedoch lassen sich an ihnen durchaus unterschiedliche Herangehensweisen betrachten. Außerdem wird ein deutliches Gewicht auf die graphische Benutzeroberfläche gelegt, um die intuitive Anwendung auch Nicht-Informatikern zu ermöglichen.

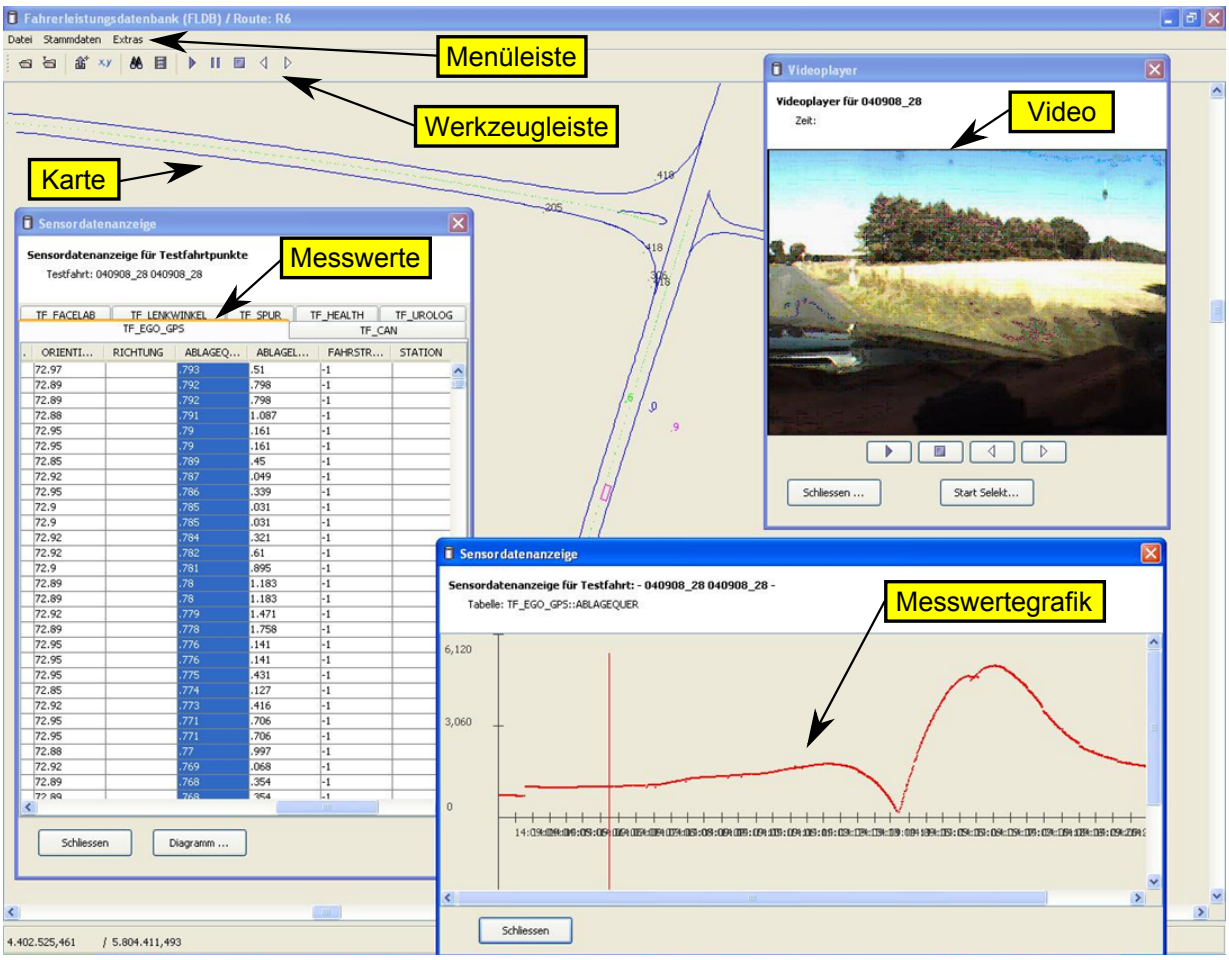


Abbildung 2-3: Graphische Oberfläche der Fahrerleistungsdatenbank[1] [Ehr07] mit Beschriftungen[5,6]

Als erster Vertreter dieser Gattung sei z.B. auf [Ten04, NT03, ET06] verwiesen, in denen die *Fahrerleistungsdatenbank* von Volkswagen vorgestellt wird. Abbildung 2-3 zeigt ein entsprechendes Bildschirmfoto. Neben einem Video und Messdaten ist auch eine Karte dargestellt. In diesem und in folgenden Bildschirmfotos kennzeichnen beschriftete Rechtecke markante Elemente. Die zugrundeliegende Idee der Fahrerleistungsdatenbank ist, das Normverhalten von Fahrern im üblichen Verkehrsablauf möglichst präzise zu dokumentieren. Hierfür werden Versuchsfahrten von Probanden dokumentiert, die auf einer sehr präzise eingemessenen Referenzstrecke stattfinden, die in digitalisierter Form mit umfangreichen Parametern (Straßengeometrie, Verkehrsschilder . . .) verfügbar ist. Diese Referenzstrecke umfasst unterschiedliche Bereiche des öffentlichen Straßenverkehrsnetzes inklusive Autobahnen, Landstraßen und Ortsbereiche. Aufgrund der engen wechselseitigen Betrachtungen zwischen Messdaten und örtlichem Bezug sind bei der Auswertung räumlich-zeitliche Fragestellungen von Interesse. Daher wurde bei der zur Datenspeicherung eingesetzten Datenbank der Gebrauch der räumlichen Erweiterungen angestrebt. Natürlich können alternative Fragestellungen bei der Datenanalyse angewendet werden, wenn z.B. Unfälle fokussiert werden (vgl.a. [Wil03]). Durch die Applikation wird keine Einschränkung der Fragestellung vorgenommen, sondern durch die Auswahl der betrachteten Versuchsfahrten.

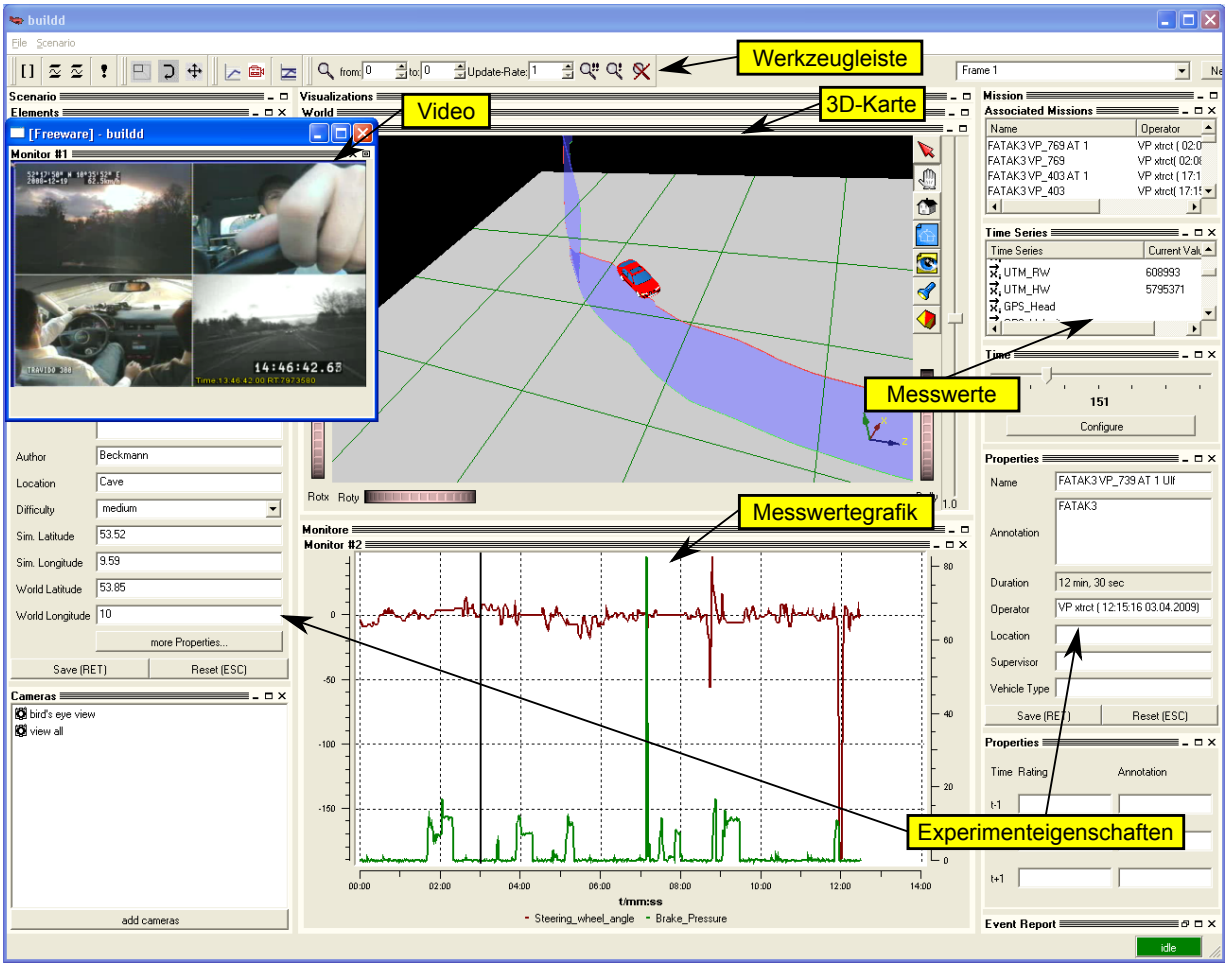


Abbildung 2-4: Graphische Oberfläche von *Ærogator*[3,5] (nach [Mey03])

Ærogator [Mey03] ist eine Anwendung zur Darstellung und Analyse von Zeitreihen. Ursprünglich wurde sie zur Betrachtung von Flugversuchen eingesetzt. Mittlerweile dient sie allerdings auch zur Untersuchung von Fahrversuchen. Im Zentrum der Betrachtungen stehen die Zeitreihen und deren Plots (Abb. 2-4), daher lässt sich diese Anwendung für andere Anwendungsfälle portieren, die sich primär auf Zeitreihen konzentrieren. Unterstützt wird der Auswerter dadurch, dass er Bedingungen auf den Zeitreihen definieren kann, die durch die Anwendung überwacht werden. Außerdem bietet *Ærogator* in einem 3D-Fenster eine Visualisierung der durch das betrachtete Fahrzeug zurückgelegten Strecke und eine Darstellung parallel aufgezeichneter Videosequenzen. Ein vergleichbares Werkzeug existiert mit Fokus auf die Bewertung von *Naturalistic Driving Studies* (NDS). Es trägt den Namen *NDS Data Analyzer* [LHP+10] und ist in Abb. 2-5 dargestellt. Die Abbildung zeigt auch, dass die Messwertevisualisierung mit Hinblick auf Fahrzeuge besonders ansprechend gestaltet ist.

In [TSGZ06, SZG+07] wird ein Werkzeug vorgestellt, das sich ebenfalls schwerpunktmäßig mit der Exploration von Zeitreihen beschäftigt und *Annotation Editor* genannt wird. Als spezielle Ausrichtung wird die semi-automatisierte Annotation von entdeckten Ereignissen und deren Abgleich mit Videos in den Vordergrund gestellt. Das Werkzeug besitzt Algorithmen zum maschinellen Lernen, um Annotationen automatisch vorschlagen zu können. So muss der Anwender nach Abschluss einer Trainingsphase die Annotationsvorschläge lediglich noch überprüfen

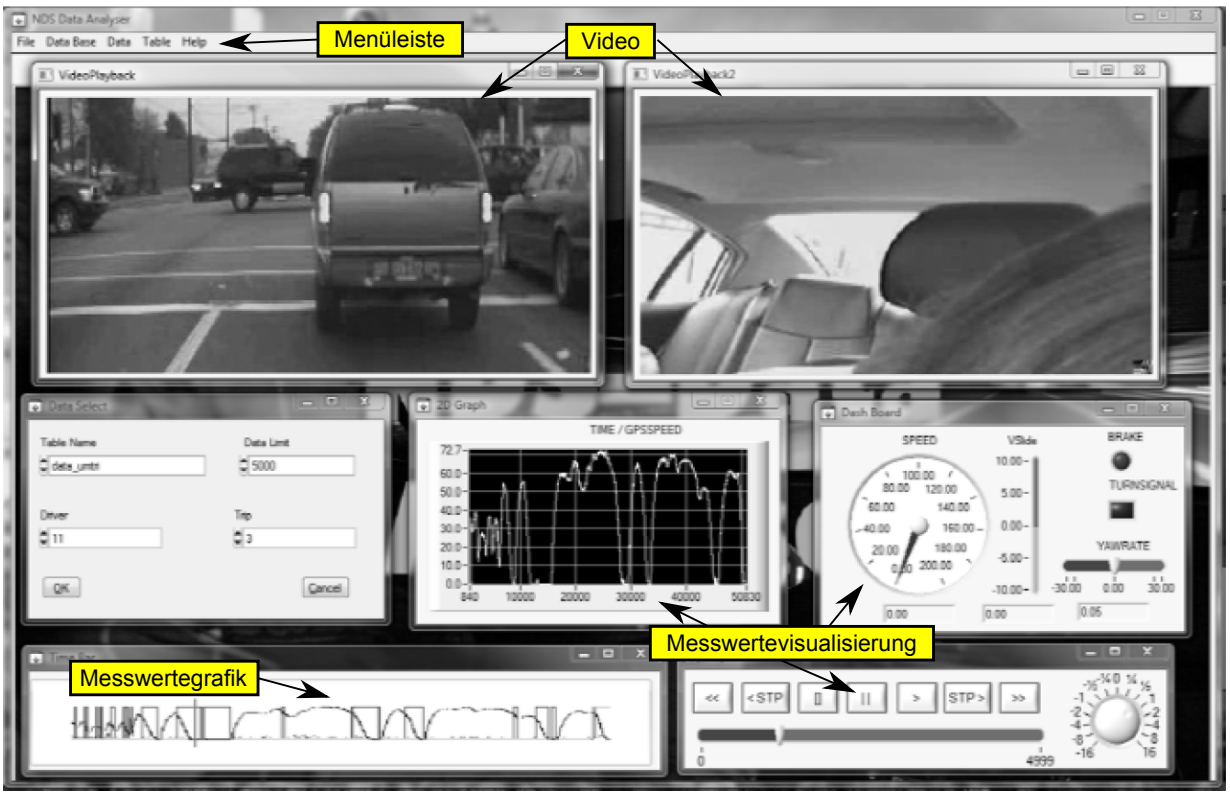


Abbildung 2-5: Graphische Oberfläche von NDS Data Analyzer[1] [LHP+10] mit Beschriftungen[5,6]

und ggf. korrigieren, was zu einer deutlichen Zeitersparnis bei der Annotation führt. Diese Eigenschaft ist für die weiteren Betrachtungen dieser Arbeit jedoch nicht relevant.

2.2.2.2 Bewertung datenbankzentrierter Informationssysteme für Fahrversuche

Ziel der folgenden Bewertung ist, die Stärken und Schwächen verwandter Arbeiten herauszustellen, um sie in Bezug zu den in dieser Arbeit behandelten Inhalten zu setzen. Die aufgeführten Informationssysteme für Fahrversuche setzen unterschiedliche Schwerpunkte in den Bereichen der Erweiterbarkeit für neue Datenreihen, Unterstützung ortsbezogener Daten, Domänenunabhängigkeit und maschinelles Lernen. Eine Übersicht über die verschiedenen Leistungsmerkmale der Anwendungen zur Exploration und Analyse von Fahrdaten ist in Tab. 2-1 dargestellt. Die rechte Spalte der Tabelle enthält eine abschließende Charakterisierung für die in diesem Kapitel behandelte Gruppe der Informationssysteme für Fahrversuche.

Bezüglich der Anwendungen lässt sich feststellen, dass datenbankzentrierte Arbeiten zum Thema Datenanalyse die Semantik der betrachteten Daten im Allgemeinen kaum berücksichtigen. Metadaten beschreiben administrative Prozesse, Sicherheitsaspekte und die Integrationsarchitektur. Sie beziehen sich nicht auf die Nutzdaten, Ausschnitte aus diesen und daraus abgeleitete Erkenntnisse.

Die vorliegenden Arbeiten enden bei der Beschreibung der Semantik der betrachteten Daten. Ohne semantische Technologien wird bei einer reinen datenbankbasierten Lösung viel Wissen implizit im Datenbanklayout versteckt, das nicht explizit zugänglich ist. Genau an diesem Punkt

	Fahrerleistungs-datenbank	Ærogator und NDS Data Analyzer	Annotation Editor	Informations-systeme für Fahrversuche
Datenverwaltung	++	++	o	++
Interpretation der Dateninhalte	++	++	++	++
Datenschema	o	++	+	+
Metadaten	o	o	o	o
Durchsuchbarkeit	+	o	+	+
Betrachtung der Datenentstehung	o	o	o	o
Behandlung orts-bezogener Daten	++	+	o	+
Benutzer-schnittstelle	++	++	++	++
Domänen-unabhängigkeit	o	o	+	o

Tabelle 2-1: Vergleich der Leistungsmerkmale von verschiedenen Informationssystemen für Fahrversuche und Bewertung dieser Anwendungsklasse

setzt die vorliegende Arbeit an und zeigt im weiteren Verlauf unter dem Einsatz aktueller semantischer Technologien neue, weiterführende Anwendungsvorschläge. Im nächsten Abschnitt wird zunächst der Entstehungsprozess von Datenprodukten betrachtet und wie dort mit Metadaten umgegangen wird.

2.2.3 Scientific Workflow Systems

Bereits seit langer Zeit werden Stapelverarbeitungssysteme bzw. Batchsysteme eingesetzt, um länger andauernde Verarbeitungsabläufe ohne Interaktion durch Computer durchführen zu lassen (vgl. [Cer03, S. 96]). Jedoch hat sich die flexible Gestaltung von Prozessabläufen in der Datenverarbeitung ständig weiterentwickelt, sodass derzeit *Scientific Workflow Systems* (SWS, [Löf12]) den Stand der Technik darstellen. Deren Schwerpunkt liegt auf der Modellierung des Datenflusses durch die Komposition verschiedener Verarbeitungsschritte. Auf Basis eines modellierten Datenflusses lässt sich eine Massenverarbeitung zukünftiger, identisch aufgebauter Daten durchführen. Der Inhalt der verarbeiteten Daten wird jedoch nicht durch das System betrachtet und als Blackbox gesehen.

Aus Sicht der vorliegenden Anwendungsdomäne lohnt es sich, diesen Ansatz ausführlich zu betrachten, jedoch lässt sich das Anwendungsszenario nur bedingt übertragen. Denn die praktische Auswertung von Fahrexperimenten ist derzeitig noch hochgradig interaktiv. Außerdem ist die Anzahl durchgeführter Versuche für in Laboren durchgeführte Studien nicht so hoch, als dass nach Etablierung einer Auswertekette für eine spezielle Studie von einer ausgeprägten Massenverarbeitung gesprochen werden könnte. Zukünftig mag sich dieser Umstand jedoch bei der Auswertung von Naturalistic Driving Studies ändern. Dies wäre der Fall, wenn die Menge der auszuwertenden Experimentalstunden um mehrere Zehnerpotenzen wächst und ggf. eine kontinuierliche, begleitende und nicht-interaktive Analyse etabliert werden muss.

Wesentliches Ziel von SWS ist die Wiederverwendungsmöglichkeit und Modularisierung der einzelnen Verarbeitungsschritte für neue Prozessketten. Der Grundgedanke von SWS ist somit dem der vergleichsweise populären Prozessmodellierung von Geschäftsprozessen mittels der *Business Process Execution Language* (BPEL, [Org07, WCL+05]) ähnlich.

Die zwei populärsten Vertreter der SWS sind B-Fabric [TP07, TAJ+10] und Kepler [ABJ+04]. B-Fabric wird an den Züricher Universitäten entwickelt und ist unter einer freien Lizenz verfügbar. Als Referenzprojekte, die B-Fabric einsetzen, werden insbesondere Projekte aus dem Bereich der Lebenswissenschaften genannt. B-Fabric stellt gleichsam die Nutzerschnittstelle zu einem großem Clustersystem dar, welches modellierte Prozesse ausführt. Experimentalsysteme werden an das B-Fabric-System angebunden, indem Ergebnisdateien über die *Secure Shell* (SSH) in das B-Fabric-System kopiert werden. Diese Dateien bilden die grundlegende Betrachtungseinheit für die Verarbeitungsschritte und können mit Metadaten und Annotationen versehen werden. Für diese Annotationen gibt es keine verbindlichen Richtlinien, jedoch sollten sie sich nach Möglichkeit an ein allgemeingültiges Vokabular halten. Gespeichert werden die Meta- und Verwaltungsdaten in einer relationalen Datenbank. Der Anwender kann über ein Webfrontend auf die Ergebnisdateien zugreifen und die entsprechenden Annotationen einsehen. Zur Suche im Datenbestand wird eine Volltextsuche mit einem Google-ähnlichen Interface angeboten. Strukturierte Abfragesprachen werden dem Endanwender nicht angeboten. In Zukunft sind für B-Fabric noch insbesondere Erweiterungen im Bereich Datenqualität, Versionierung und Data Lineage angedacht.

Das Kepler-System [ABJ+04] setzt seinen Schwerpunkt insbesondere auf Aspekte des Datenflusses zur Erzeugung von Datenprodukten. Kepler steht unter der *Berkeley Software Distribution* (BSD) Open-Source-Lizenz zur Verfügung und baut dabei auf dem Vorgänger *Ptolemy II* auf. Zur anschaulichen Visualisierung und graphischen Bearbeitung steht ein Editor zur Verfügung. In diesem wird der Datenfluss als gerichteter Graph dargestellt, bei dem die Knoten einzelne Verarbeitungsschritte darstellen und als Blöcke visualisiert werden. Gespeichert werden diese Graphen als XML-Datei, die dem Ausführungskern übergeben werden, der die Koordination des Datenflusses übernimmt. Wiederverwendbare Datenflusselemente können Quellen, Senken, Transformationen, analytische Schritte oder sonstige Berechnungen sein. Jedes dieser Elemente kann mehrere Ein- oder Ausgangs-Pins besitzen und außerdem noch mit einer statischen Parametrisierung versehen werden. Kepler stellt sicher, dass typkompatible Pins miteinander verknüpft werden. Die Ausführung der Verarbeitungsschritte ist dann in einer verteilten Umgebung möglich. Insgesamt wird offensichtlich, dass eine Parallelität zu Webservices [BHM+12] besteht. So haben die Entwickler letztendlich die Möglichkeit geschaffen, Webservices, welche zur *Web Services Description Language* (WSDL) konform sind, als Verarbeitungsschritte aufzurufen. Kernelemente der Verarbeitung sind genau wie bei B-Fabric wieder Dateien, wobei der Zugriff über das *File Transfer Protocol* (FTP) ermöglicht wird.

Zwei weitere bekannte SWS sind *Triana* [Car09, HWTS08] und *Taverna Workbench* [myG09, OAF+04], welche als Open-Source-Software unter der Apache-Lizenz (Version 2.0) und der *Les-*

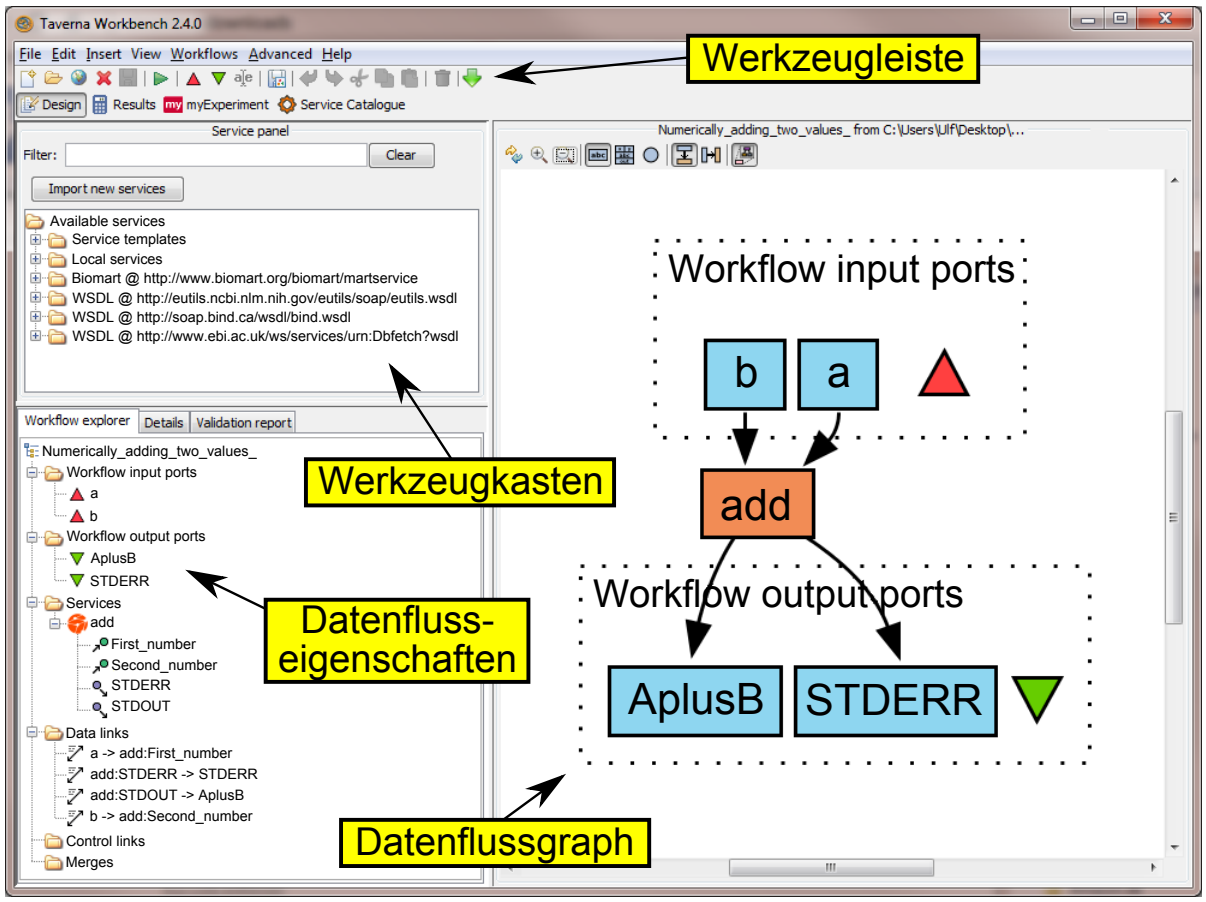


Abbildung 2-6: Graphische Oberfläche von Taverna zur Bearbeitung von Datenflussgraphen[3,4,5] (nach [myG09], Lizenz: [Fre99])

ser General Public License (LGPL, Version 2.1, [Fre99]) stehen. Triana und Taverna sind beide stark an Grid-Prinzipien ausgerichtet und Taverna wird außerdem in einem großen Grid-Projekt von mehreren Universitäten weiterentwickelt. Abbildung 2-6 zeigt die Benutzeroberfläche von Taverna, die typisch für ein SWS ist. Mit Hilfe des Datenflussgraphen modelliert der Anwender den abzuarbeitenden Prozessablauf. In verschiedenen Fenstern sind die Datenflusseigenschaften und die zur Verfügung stehenden Werkzeuge dargestellt.

Darüber hinaus existieren im *Deutschen Zentrum für Luft- und Raumfahrt* (DLR) eingesetzte SWS-Entwicklungen, die in der Öffentlichkeit nicht sonderlich bekannt sind. Dies sind die Prozesssysteme *Data Information and Management System* (DIMS, [MDG+00, BRMR01, KMG01]) und DataFinder [SS07, Sch09b]. DIMS wird am *Deutschen Fernerkundungsdatenzentrum* (DFD) entwickelt und dort seit dem Jahr 2000 zur Erstellung, Archivierung und Verteilung von Produkten der Erderkundung in mehreren Missionen eingesetzt. In Veröffentlichungen wird DIMS als *Middleware* zur Integration von Datenverarbeitungssoftware in die DIMS-Arbeitsumgebung beschrieben (vgl. [BRMR01]). Über eine *Programmierschnittstelle* (*Application Programming Interface*, API) können Datenverarbeitungsprogramme in die DIMS-Arbeitsumgebung eingebunden werden, sodass die Anwendungen über diese ihre Eingaben beziehen und ihre berechneten Ausgaben abliefern. Der Prozessfluss als die Verarbeitungsabfolge der eingebundenen Applikationen lässt sich über ein externes Regelwerk definieren. Es können Trigger und Timer eingebun-

den werden, um eine vollständig automatisierte Ausführung des Datenproduktionsprozesses zu ermöglichen. DIMS übernimmt außerdem bei der Prozessteuerung eine Lastverteilung auf die Computer des Rechenclusters, sodass in vielen Fällen das System alleine schon durch das hinzufügen neuer Computer skaliert. Grundlegende Betrachtungseinheit für die Verarbeitung sind wie bei den anderen aufgeführten SWS ebenfalls Dateien und Verzeichnisse. Für diese existieren kleine XML-Dateien, die allgemeine Metadaten (z.B. Erfassungszeitpunkt, Aufzeichnungsort etc.) beschreiben. Sobald die Dateien in DIMS archiviert sind, werden die Metadaten für spätere Suchabfragen in einer Datenbank gespeichert. Der Austausch der Dateien für weitere Verarbeitungsschritte erfolgt mittels FTP und wird über die DIMS-API gesteuert. Somit wird durch DIMS von der verwendeten Speichertechnologie abstrahiert. Im Hintergrund arbeitet die *Product Library*, die die langfristige Speicherung der Dateien übernimmt, sie bei Bedarf wieder zur Verfügung stellt und sie anhand der Metadaten katalogisiert. So kommt ein hierarchisches Speichersystem zum Einsatz, bei dem die Dateien bei nur seltener Nutzung in einem Bandarchiv mit Bandroboter zur automatischen Bestückung gespeichert werden. Zugriff auf die durch DIMS gespeicherten Elemente wird über eine Webschnittstelle ermöglicht. Es lässt sich festhalten, dass DIMS eine sehr weitgehende Automatisierung bis hin zur Archivierung ermöglicht, um über sehr lange Zeiträume Datenerfassungen ohne manuelle Interaktion zu ermöglichen. So lassen sich mehrjährige Satellitenmissionen unterstützen, bei denen kontinuierlich Daten erfasst und für einen langen Zeitraum archiviert werden müssen.

Eine zweite im DLR eingesetzte SWS-Anwendungssoftware ist *DataFinder* [SS07, Sch09b], welche unter einer BSD-Lizenz verfügbar ist. Ihr Aufgabenschwerpunkt ist die Verwaltung großer technisch-wissenschaftlicher Daten und dabei von den verschiedenen Speichertechniken zu abstrahieren, wobei auf offene und flexible Standards gesetzt wird. Als Speicherschnittstellen werden *Web-based Distributed Authoring and Versioning* (WebDAV), FTP, GridFTP, *Open Andrew File System* (OpenAFS) und *Tivoli Storage Manager* (TSM) unterstützt. Die Hauptfunktionalitäten des DataFinders sind der Up- and Download von Dateien, das Versehen der Daten mit Metainformationen, eine Suchfunktion und die Abarbeitung von Skripten zur Automatisierung von Berechnungen und Arbeitsabläufen. Zusätzlich zu vorgegebenen Metadaten-Attributen können diese um benutzerdefinierte erweitert werden, um so die Beschreibung und später die Suche zu unterstützen. Durch die Verwendung von WebDAV als Speichertechnologie wird die Versionierung von Inhalten unterstützt. Aktuell wird DataFinder insbesondere für den Einsatz in Grid-Szenarien optimiert. Im Gegensatz zu den anderen bisher betrachteten SWS wird bei DataFinder kein so großer Schwerpunkt auf Betrachtungen zum Datenfluss gelegt. Insgesamt ist DataFinder somit eine leichtgewichtige Lösung, die sich einfach zur Verwaltung von Datenbeständen einsetzen lässt, aber nur eine begrenzte Unterstützung und keine innovativen Konzepte für Metadaten bietet.

In [HWAK08, WHA+09, WK09] wird ein Metadaten-Framework zur Integration semantisch heterogener Wissenspools vorgestellt. Grundlage für die vorgestellten Betrachtungen ist, dass, wie bei SWS üblich, Metadaten auf Datei-Ebene vergeben werden. Traditionell wird wie in den bisher behandelten SWS-Lösungen für Metadaten ein Katalogsystem von Schlagworten, das

	B-Fabric, Kepler, Taverna und Triana	DIMS	DataFinder	Scientific Workflow Systems
Datenverwaltung	o	+	+	+
Interpretation der Dateninhalte	o	o	o	o
Datenschema	o	o	o	o
Metadaten	o	o	+	o
Durchsuchbarkeit	o	o	+	o
Betrachtung der Datenentstehung	++	++	++	++
Behandlung ortsbezogener Daten	o	+	o	o
Benutzerschnittstelle	+	+	+	+
Domänenunabhängigkeit	++	++	++	++

Tabelle 2-2: Vergleich der Leistungsmerkmale von verschiedenen Scientific Workflow Systems und Bewertung dieser Anwendungsklasse

erweiterbar ist, eingesetzt. Das in diesen Arbeiten vorgestellte Metadaten-Framework führt ein Katalogsystem basierend auf Ontologien ein, um die zu archivierenden Dateien zu katalogisieren. Hintergrund dieser Veröffentlichungen ist es, die graphische Benutzerschnittstelle in Abhängigkeit von den Attributen der betrachteten Ontologie entsprechend anzupassen. Jedoch ist das vorgestellte Metadaten-Framework zum Zeitpunkt der Publikationen noch nicht in ein SWS oder eine Grid-Software integriert und wird somit noch nicht aktiv eingesetzt.

Ein weiterer Aspekt, der in Zusammenhang mit SWS Erwähnung findet, ist die Historie der verarbeiteten Daten. Diese wird als *Provenance* oder *Lineage* bezeichnet. Zu diesem Themengebiet gibt es zahlreiche eigenständige Veröffentlichungen (z.B. [DBE+07, Bos02, BKT01, CJ07, Tan07]). Hierbei stehen die in einem SWS durchgeführten Transformationsschritte im Fokus und werden weiter untersucht. Somit werden verschiedene Transformationstypen, deren Kombinationen und ihre Auswirkungen genau klassifiziert. Weiterhin ist die Dokumentation über die durchgeführten Verarbeitungsschritte von Interesse. Ähnliche Betrachtungen werden für DWH-Systeme angestellt (Abschnitt 2.2.1).

2.2.3.1 Bewertung von Scientific Workflow Systems

Für die SWS lässt sich insgesamt festhalten, dass Betrachtungen und Metadaten zum Datenverarbeitungsprozess und nicht die Analyse selbst im Vordergrund stehen. Die zu verarbeiteten Datenpakete werden jedoch immer als separate Dateien zwischen den Verarbeitungsschritten im System übergeben. Die Dateien werden komplett als Blackbox betrachtet und immer nur mit allgemeinen Metainformationen versehen. Eine Bewertung der Leistungsmerkmale der einzelnen Systeme ist in Tab. 2-2 dargestellt und für die gesamte Anwendungsklasse in der rechten Spalte zusammengefasst.

Genau an diesem Punkt geht die vorliegende Arbeit einen Schritt weiter. Bei den folgenden Betrachtungen wird der Blick auf den Inhalt der Dateien in den Fokus gesetzt, um so die Auswertung durch Metadaten auf dieser Ebene zusätzlich zu unterstützen. Zunächst wird im folgenden Abschnitt betrachtet, wie Datenprodukte in spezialisierten Datenhaltungssystemen verwaltet werden und Metadaten bei dieser Aufgabe unterstützen.

2.2.4 Wissenschaftliche Datenhaltungssysteme

Wissenschaftliche Datenhaltungssysteme haben die Aufgabe, wissenschaftliche Mess- und Versuchsdaten zu archivieren und auf Abruf bereitzustellen. Einige SWS weisen ebenfalls diese Merkmale auf (z.B. DIMS oder DataFinder), welche sowohl die Datenverarbeitung als auch die Archivierung betrachten. Im Folgenden werden jedoch Beispiele reinrassiger Datenhaltungssysteme betrachtet. Diese Datenhaltungssysteme setzen zur Beschreibung der archivierten Daten ebenfalls Metadaten ein, die durchsucht werden können. Alle betrachteten Systeme sind als Webportal realisiert, sodass sie einem großen Publikum zugänglich sind. Über sie können so in internationalen Teams Datensätze ausgetauscht werden. Insbesondere in der Erdsystemforschung und Umweltbeobachtung werden die folgenden Systeme eingesetzt.

Abbildung 2-7 zeigt Bildschirmfotos von *PANGAEA — Data Publisher for Earth & Environmental Science* [Alf14a, DGR+02], welches vom Alfred-Wegener-Institut betrieben wird. Die Dokumentation des Systems erfolgt über ein separates Wiki [Alf14b]. In Abb. 2-7(a) ist die Homepage zu sehen, welche eine Suche in den Datenbeständen ermöglicht. Über das auch als Suchschlitz bezeichnete Eingabefeld lassen sich in der Such-Syntax von Google Stichworte (hier: „Carbon Isotope Northern Hemisphere“) eingeben, nach denen die Metadaten durchsucht werden. Die Abb. 2-7(b) zeigt einen archivierten Datensatz, der über die Suche gefunden wurde. In der oberen Hälfte werden die fest definierten Metadaten dargestellt und in der unteren Hälfte können die hinterlegten Datensätze als Datei heruntergeladen werden. Metadaten werden grundsätzlich nach einem definierten Schema erfasst, wobei die meisten Angaben in Freitext erfolgen und auch Geokoordinaten mit abgefragt werden. Die Metadaten können in verschiedene Formate exportiert werden, wobei auch Dublin Core ([Dub09], vgl. Abschnitt 2.3.7) unterstützt wird. Das *Biodiversity-Exploratory Information System* (BExIS, [Sch10, HNKR10]) verfolgt einen vergleichbaren Ansatz wie PANGAEA, ist jedoch nicht öffentlich zugänglich.

In der *Global Biodiversity Information Facility* (GBIF, [Kin10, Edw04]) werden Organismen und deren Auftreten archiviert. Es existiert eine Ordnung der Lebewesen, wobei für jedes Orte gespeichert werden, an denen diese gesichtet wurden. Für jede Sichtung existiert ein fest definierter Satz an Metadaten. Da ein sehr hoher Positionsbezug besteht, ist eine Abfrage über Karten möglich, bei der auf einer Karte Quadranten ausgewählt werden können. Lokale und nationale Biodiversitätsdatenquellen werden durch GBIF über Webservices integriert.

Das *International Long Term Ecological Research Network* (ILTER, [ILT10, Mag90]) ist ein loser Verbund von nationalen Einrichtungen zur Sammlung von Umweltmessdaten. Zum einen

(a) Suchportal

(b) Ergebnis (Kartendaten: Google, NASA)

Abbildung 2-7: PANGAEA — Data Publisher for Earth & Environmental Science[3,4,5,6] (nach [Alf14a])

wird eine Katalogisierung von Publikationen ermöglicht. Zum anderen werden für das europäische Netz durch die nationalen Einrichtungen teilweise ebenfalls Datenbanken mit Messdaten bereitgestellt (vgl.a. [MKS10]). Zur Datenintegration der verschiedenen nationalen Datenquellen wird eine gemeinsame Ontologie eingesetzt, sodass ein Schema-Mapping der einzelnen Datenbanken auf ein Zielschema ermöglicht wird (vgl. Ansatz von [Köh03, KES+07]). Metadaten werden auf der Ebene von Datentabellen nach einem festen Schema erfasst (vgl. [MBHaSGS97, Mic06]).

Bei dem *Massachusetts Ocean Resource Information System* (MORIS, [Coa10]) handelt es sich um ein klassisches Geoinformationssystem [Bun07] zur Verwaltung von ozeanographischen Messwerten. Die Architektur basiert auf einem Datenbank-Backend, einer Geoinformations-Middleware und einem Frontend-Server, der die Daten in Karten einzeichnet und als Webseiten

	PANGAEA und BExIS	GBIF	ILTER	MORIS	Wissenschaftliche Datenhaltungssysteme
Datenverwaltung	+	+	+	+	+
Interpretation der Dateninhalte	o	o	o	o	o
Datenschema	o	o	o	o	o
Metadaten	+	o	++	o	+
Durchsuchbarkeit	++	+	+	++	++
Betrachtung der Datenentstehung	o	o	o	o	o
Behandlung ortsbezogener Daten	+	++	+	++	++
Benutzerschnittstelle	+	+	+	+	+
Domänenunabhängigkeit	+	o	+	+	+

Tabelle 2-3: Vergleich der Leistungsmerkmale von verschiedenen wissenschaftlichen Datenhaltungssystemen und Bewertung dieser Anwendungsklasse

ausliefert. Weiterhin erlaubt der Frontend-Server eine Bereitstellung in ein in der Geoinformatik verbreitetes Format für Kartenelemente, sodass die georeferenzierten Elemente in eigene Kartendarstellungen übernommen werden können. Eine Verwaltung sämtlicher Daten und Metadaten erfolgt somit ausschließlich über die explizite Modellierung in der zugrundeliegenden Datenbank.

2.2.4.1 Bewertung wissenschaftlicher Datenhaltungssysteme

Sämtliche vorgestellten und bekannten Datenhaltungssysteme ermöglichen die Archivierung und den Zugriff auf Mess- und Versuchsdaten. Der Fokus liegt hierbei insbesondere auf einer besonders leichten Durchsuchbarkeit durch die Anwender. Die hinterlegbaren Metadaten beziehen sich jedoch bisher nie auf einzelne feingranulare Messdatenelemente. Für die Metadaten werden nur sehr rudimentär oder gar nicht semantische Technologien verwendet, sodass immer nur ein fest vorgegebener Satz von Metadaten-Attributen erfasst werden kann. Daher besteht kaum Interoperabilität mit fremden Anwendungsszenarien. Eine detaillierte Zusammenfassung und Bewertung der einzelnen Leistungsmerkmale ist in Tab. 2-3 dargestellt. Die rechte Spalte enthält die zusammenfassende Bewertung der betrachteten Anwendungsklasse.

Im weiteren Verlauf zeigt diese Arbeit, wie semantische Technologien eingesetzt werden können, um die genannten Probleme zur Verwaltung von wissenschaftlichen Daten zu umgehen. Durch den formalen Charakter der eingesetzten Technologien werden komplexe Suchabfragen für Messdatenelemente ermöglicht. Außerdem erlaubt deren Interoperabilität die Integration verschiedener Systeme. So wird beispielsweise gezeigt, wie ein semantisches Wiki zur umfassenden Dokumentation eingebunden werden kann. Der nächste Abschnitt stellt zunächst semantische Technologien zur Beschreibung feingranularer Datenelemente aus anderen Bereichen vor, sodass die sich daraus ergebenden Möglichkeiten erkennbar werden.

2.2.5 Inhaltliche Indizierung und Annotation

Im Gegensatz zu den SWS liegt bei den Ansätzen zur semantischen Indizierung der zentrale Blickpunkt nicht mehr auf dem Prozess, sondern auf den Daten selbst. Das Ziel ist es, den wesentlichen Inhalt und die Bedeutung der Daten so zu beschreiben, dass sie maschinell weiterverarbeitbar sind. Die Daten werden somit nicht mehr als Blackbox betrachtet. Wesentlichen Schwung gewonnen hat dieser Ansatz mit der Entstehung der Vision des Semantic Webs und der hierfür eingesetzten Technologien. Dementsprechend fokussieren sich Arbeiten aus diesem Themenbereich insbesondere auf die Beschreibung von Webseiten und deren Inhalten. Der Großteil dieser Bestrebungen bezieht sich somit auf die Beschreibung von Texten. Weiterhin ist die Behandlung von Multimediadaten wie Bildern, Videos und Musik aktiver Gegenstand der Forschung und bereits in Anwendungen zu finden. Die Beschreibung von Messdaten oder allgemein numerischer Rohdaten in Tabellenform, ist jedoch nicht verbreitet. Daher soll im Rahmen der vorliegenden Arbeit gezeigt werden, dass diese Prinzipien für die Verarbeitung von Messdaten praktikabel sind.

Bei der klassischen Indizierung von Datenbeständen werden die wesentlichen Elemente des betrachteten Objektes extrahiert und zusammen mit ihrem Bezug zum ursprünglichen Objekt in einem Index gespeichert. Die Struktur des Indexes wird dabei so gewählt, dass er möglichst schnell und effizient durchsucht werden kann. Wird ein Suchbegriff vorgegeben, können sämtliche Datensätze und ggf. die relevanten Stellen schnell herausgesucht werden. Auf dieser Basis mittels indizierter Stichwörter arbeiten Internetsuchmaschinen in einer sehr großen Dimension. Im Gegensatz zu den SWS-Ansätzen werden die Datensätze nicht mehr als Blackbox betrachtet, sondern deren Inhalt untersucht. Für diese Vorgehensweise ist jedoch immer ein grundlegendes Verständnis über den Aufbau der betrachteten Datensätze notwendig. So muss eine Internetsuchmaschine den Inhalt eines Dokumentes von den Schlüsselwörtern der *Hypertext Markup Language* (HTML) für die Textformatierung einer Internetseite unterscheiden. Für die Indizierung des Inhalts sind die HTML-Anweisungen irrelevant. Sollen beispielsweise Dokumente von *Microsoft Word* oder im *Portable Document Format* (PDF) indiziert werden können, gelten natürlich die gleichen Voraussetzungen.

Aktuelle Forschungsarbeiten beschäftigen sich damit, die Semantik der betrachteten Inhalte mit zu berücksichtigen. Denn bei einer Suchanfrage nach „Kohl“ erwartet man unterschiedliche Ergebnisse, je nachdem ob man ein Gemüse oder eine Person meint. Die Herausforderung liegt somit zum einen in der Erfassung der Semantik als auch in deren Verarbeitung und Speicherung. Um die Semantik großer Textmengen automatisiert zu extrahieren, werden weitgehend automatisierte Verfahren eingesetzt, die unter dem Begriff *Text Mining* zusammengefasst werden. Sowohl hier als auch in anderen Bereichen etablieren sich semantische Technologien auf Basis des *Resource Desciption Frameworks* (RDF) und der *Web Ontology Language* (OWL) zur Beschreibung der gewonnenen semantischen Zusammenhänge. In Abschnitt 2.3 werden diese ausführlich vorgestellt. In diesem Zusammenhang wird von Annotationen gesprochen,

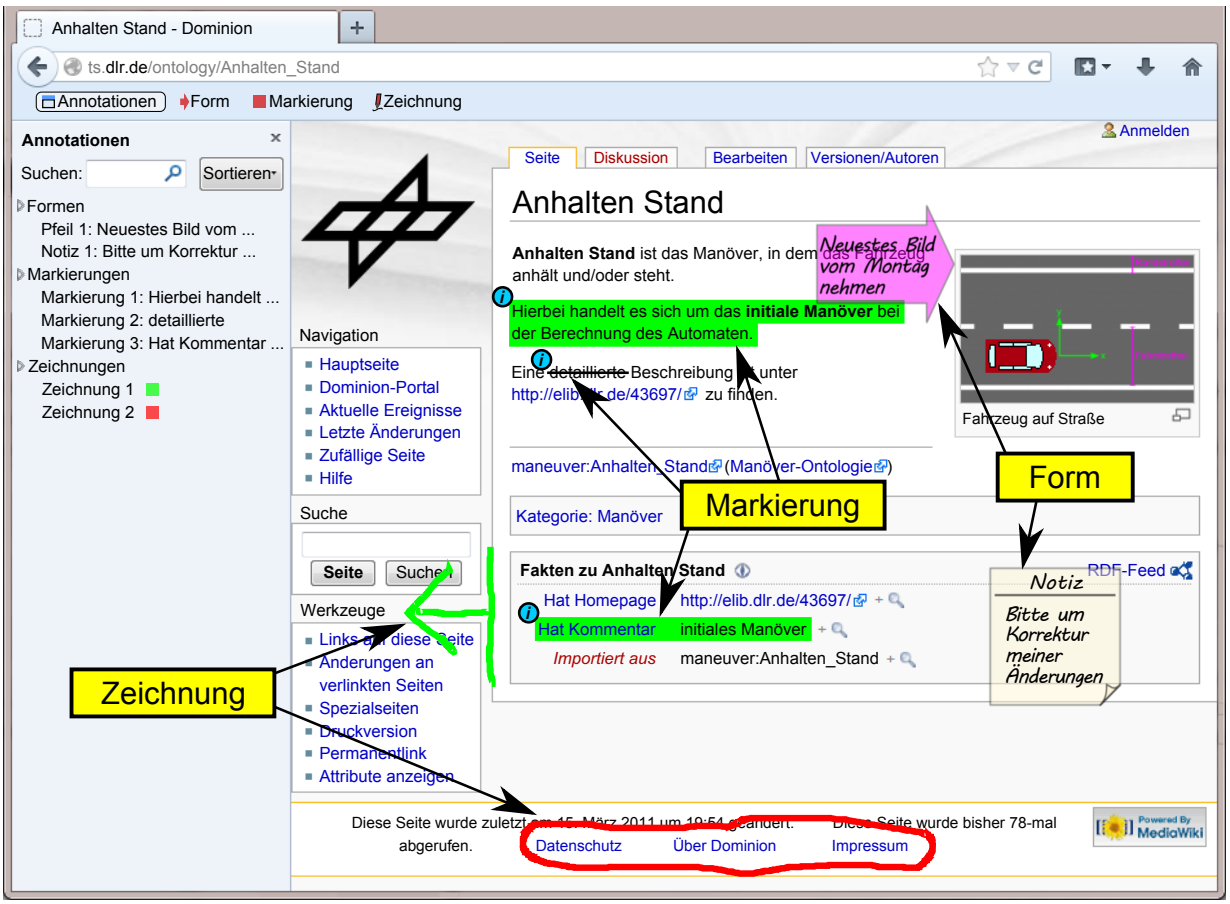


Abbildung 2-8: Prinzip der Annotation von Webseiten in einer Bildmontage[4,5] (frei nach [BL09, KKPS01])

wenn vorhandene Informationen mit zusätzlichen Informationen über deren Bedeutung versehen werden. Je nach Typ der zu betrachtenden Datensätze gibt es unterschiedliche Projekte, die sich mit deren Indizierung befassen. So sind sehr verschiedene Vorgehensweisen anzutreffen, um die Semantik der Elemente zu bestimmen — von der manuellen Annotation bis hin zu spezialisierten Algorithmen für einen Anwendungsfall.

Im Folgenden werden einige Anwendungen zur Annotation von Webseiten vorgestellt, die es ermöglichen, den Inhalt dieser Seiten zu beschreiben. Die Software *iMarkup* [BL09] ermöglicht es dem Anwender, ähnlich einer Zeichenapplikation graphische Elemente der in einem Webbrowser dargestellten Webseite hinzuzufügen. Dabei kann er zwischen freiem Zeichnen, Formen und Markierungen hinzufügen oder Texte eingeben wählen und dabei diese Elemente in verschiedenen Farben hervorheben. Die Bildmontage in Abb. 2-8 zeigt das Prinzip der Annotation von Webseiten in Anlehnung an iMarkup.

Das Projekt *Annotea* [KKPS01] des World Wide Web Consortiums wurde im Rahmen der Semantic Web Bewegung gegründet (Abschnitt 2.3) und basiert auf ähnlichen Prinzipien wie iMarkup, geht dabei jedoch über dessen Möglichkeiten hinaus. Ziel ist es, die Zusammenarbeit verschiedener Anwender mittels gemeinsam verwendbarer und auf Metadaten basierender Web-Annotationen, Web-Lesezeichen und deren Kombination zu unterstützen. In diesem Zusammenhang werden unter Annotationen Kommentare, Notizen, Erklärungen und andere Typen von externen Anmerkungen verstanden, die sich auf eine Webseite oder ausgewählte

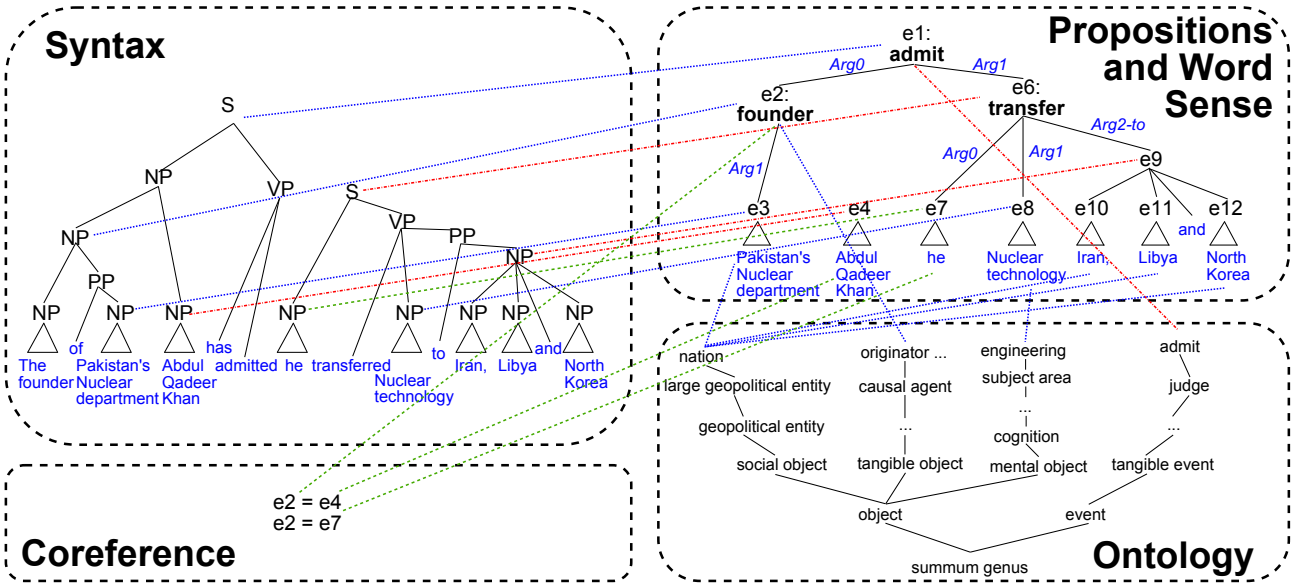


Abbildung 2-9: Satzdekomposition in OntoNotes[2] (nach [Ray11, WPR+12])

Abschnitte dieser beziehen. Erstellt werden sie durch manuelles Bearbeiten durch den Autor einer Webseite oder einen beliebigen Besucher. Hierfür ist es nicht notwendig, das eigentliche Dokument zu verändern. Bei der Realisierung dieser Ziele soll soweit wie möglich auf bestehende Standards des World Wide Web Consortiums aufgebaut werden. Es wird ein auf RDF basierendes Schema für die Modellierung der Metadaten verwendet und zur Referenzierung von Teilelementen wird auf die *XML Pointer Language* (XPointer, [DMD01]) und die *XML Linking Language* (XLink, [DMO01]) zurückgegriffen. Die Verwaltung der Lesezeichen wird mit analogen Ansätzen realisiert. Zur Speicherung der Daten kann entweder der eigene Rechner oder ein Server gewählt werden, um diese mit anderen Anwendern zu teilen. Als Übertragungsprotokoll zum Server wird das standardisierte *Hypertext Transfer Protocol* (HTTP) eingesetzt.

Ein sehr ähnlicher Ansatz wird in den Arbeiten von Gertz [GS01, GS02a] aufgezeigt, bei dem Webseiten mit wissenschaftlichem Inhalt beschrieben werden. Dabei wird insbesondere ein Anwendungsfall aus den Neurowissenschaften betrachtet. Der Fokus wird auf die Gesamtarchitektur als verteiltes System für eine kollaborative Zusammenarbeit gelegt. Es wird jedoch kein RDF für die Annotationen angewendet, sondern eine eigene, ähnliche und deutlich primitivere Sprache. Für diese wird eine Transformation in SQL-Ausdrücke auf einem definierten Datenbankschema vorgestellt, sodass in den Annotationen sehr effizient gesucht werden kann.

In diesem Zusammenhang sei auf das OntoNotes-Projekt [HMP+06, Ray11] verwiesen. Dessen Ziel ist die Modellierung der semantischen Informationen ganzer Wörterbücher mittels Ontologien und entsprechenden semantischen Technologien (Abschnitt 2.3). Die inhaltliche Zerlegung eines Satzes ist in Abb. 2-9 dargestellt. Der dargestellte Satz wird sowohl syntaktisch als auch auf Ebene der getroffenen Aussage betrachtet und die Elemente auf eine Ontologie abgebildet. Dabei werden die Sprachen Englisch, Chinesisch und Arabisch behandelt. Somit sind Verweise zwischen den verschiedenen Sprachen möglich. Anwendungsziel ist der Einsatz für sprachliche Applikationen und im Bereich der automatisierten Analyse von Texten und deren automatisierter

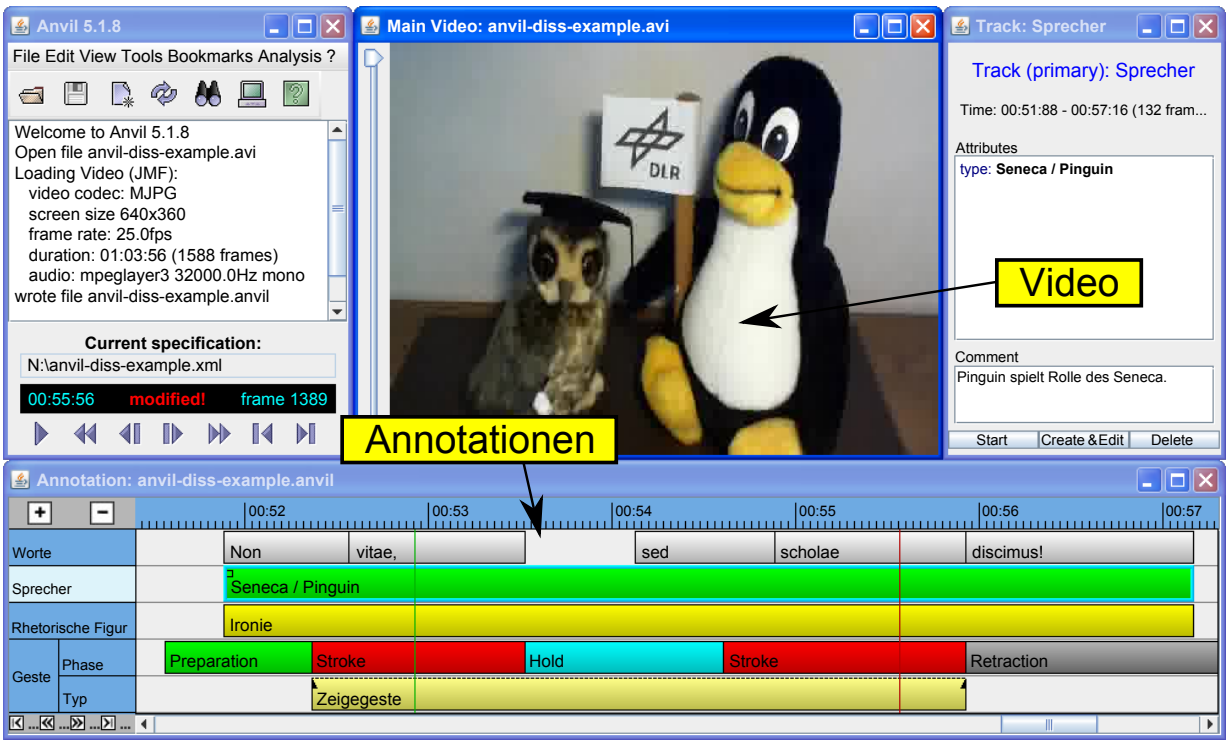


Abbildung 2-10: Videoannotation mit Anvil[3,4,5] (nach [Kip01])

Annotation. Ein nicht ganz so weit gehender Ansatz wird im WordNet-Projekt [Fel98] verfolgt. Es fasst Synonyme englischer Nomen, Verben und Adjektive in erkenntnismäßige Gruppen zusammen, welche jeweils ein Konzept repräsentieren.

Die für Webseiten beschriebenen Ansätze werden in einer vergleichbaren Form zur Beschreibung von Multimediadaten und Videos eingesetzt. Zur automatisierten Extraktion von Inhalten aus Videos werden bei Bedarf Techniken der Bildverarbeitung eingesetzt, was für diese Arbeit jedoch nicht von Bedeutung ist. Die Aufgabe dieser Anwendungen ist, den Inhalt von Videos zu codieren. Dabei können sowohl die auf den Standbildern sichtbaren Gegenstände, als auch die Bedeutung von Sequenzen beschrieben werden. Dieser Ansatz zur Annotation von Videos ist beispielsweise auch relevant für die Auswertung von Aufnahmen aus Fahrversuchen und Naturalistic Driving Studies.

In [Kip01] wird das Videoannotationswerkzeug *Anvil* vorgestellt, welches unter einer freien Lizenz verfügbar ist. Es erlaubt das framegenaue Codieren von Videos mit einem benutzerdefiniertem Schema. In der unteren Hälfte von Abb. 2-10 sind die Codierungen in mehreren Spuren farblich in einer zeitlichen Balken-Übersicht dargestellt. Außerdem kann das Tool mit Werkzeugen zur Sprachannotation wie *XWaves* oder *Praat* [vH01] zusammenarbeiten, die eine ähnliche Vorgehensweise für die Sprachanalyse ermöglichen. Zur Speicherung wird ein auf XML basierendes Dateiformat verwendet.

Eine weitere Anwendung zur Codierung von Videoinhalten sei mit dem *ProjectPad VideoTool* [Nor09, CRPC09] erwähnt. Insgesamt gibt es jedoch für diesen Zweck zahlreiche freie und kommerzielle Anwendungen, sodass eine umfassende Auflistung nicht möglich und auch nicht sinnvoll ist.

Ebenfalls wird in dem Bereich der Videocodierung an der Einführung von semantischen Technologien für Annotationen gearbeitet. So wird in [BPS+05] die Beschreibung von Elementen aus Videos im Format *Moving Picture Experts Group 7* (MPEG-7) mittels einer Ontologie durchgeführt. Dabei wird inhaltlich sowohl auf einer sehr rudimentären Ebene gearbeitet (z.B. dominante Farbe) und zum anderen auf einer abstrakteren, in der z.B. Personen oder Fahrzeuge abgebildet werden. Zu diesem Zweck wird eine selbst entwickelte Software *M-OntoMot-Annotizer* eingesetzt.

Zur Annotation von Musik wird außerdem in [HOCM+05] mit *A music content semantic annotator* (MUCOSA) eine erwähnenswerte Arbeit vorgestellt. Dabei handelt es sich um eine Drei-Schichten-Architektur zum verteilten Verarbeiten von Musikstücken. Es besteht zum einen aus einem Client für Mikro-Annotationen, die sich auf Ausschnitte innerhalb eines Musikstückes beziehen. Auf Ebene der Mikro-Annotationen ist insbesondere die semantisch, inhaltliche Ebene der Lieder von Bedeutung. Zum anderen existiert ein Client für Makro-Annotationen. Die bearbeiteten Makro-Annotationen beziehen sich auf ganze Musikstücke. Der Editor besitzt außerdem Komfortfunktionen, die es vereinfachen, die Annotationen für ganze Musiksammlungen zu verwalten. Als grundlegende Ebene für beide Editoren wird ein Annotationssubsystem verwendet. Eine weitere Rolle spielt ein Teilprojekt zum (semi-)automatisierten Generieren von Annotationen. So erstellte Annotationen dienen als Template oder Vorschlag für das manuelle Bearbeiten. Für das gemeinsame Arbeiten werden Rollen definiert, die jeweils unterschiedliche Rechte besitzen und somit verschiedene Aufgaben wahrnehmen. Die von den verschiedenen Bearbeitern erstellten Annotationen werden durch das zentrale Subsystem auf einem Server synchronisiert. Eine ähnliche Client-/Server-Architektur für das verteilte Annotieren von Videos auf Basis semantischer Grundlagen wird in [SHK03] diskutiert.

Weiterhin existiert eine weitergehende Literatur-Übersicht in [Asa08]. Inhaltlich konzentriert sich die Übersicht auf Metadaten-Annotationen und deren Verknüpfung mit verschiedenen Datenmodellen. Für die Annotationen werden ebenfalls semantische Technologien diskutiert. Zuerst werden diese Betrachtungen allgemein durchgeführt, um sich dann auf Webseiten als Ziel von Annotationen zu fokussieren.

2.2.5.1 Bewertung inhaltlicher Indizierung und Annotation

Zum Thema Annotation lässt sich festhalten, dass dies für Webseiten und für Mediadaten, wie Videos und Musik, einen etablierten Ansatz darstellt. Insbesondere für Webseiten werden semantische Technologien auf Basis von RDF eingesetzt (Abschnitt 2.3). Da diese Technologien ursprünglich in diesem Anwendungsbereich entstanden sind, ist dies leicht nachvollziehbar. Im Laufe der Zeit hat dann eine entsprechende Adaption von Annotationen auf Multimediainhalte und andere Bereiche stattgefunden.

Die Struktur bzw. das Schema der Nutzdaten, die Datenentstehung und deren Ortsbezug spielen dagegen keine Rolle. In Tab. 2-4 werden die Leistungsmerkmale für die verschiedenen An-

	Web-Annotation (z.B. Annotea)	Text-Annotation (z.B. OntoNotes)	Video-Annotation (z.B. Anvil)	Audio-Annotation (z.B. MUCOSA)	Inhaltliche Indizierung und Annotation
Datenverwaltung	+	o	o	+	+
Interpretation der Dateninhalte	++	++	+	+	++
Datenschema	o	+	o	o	o
Metadaten	++	++	++	+	++
Durchsuchbarkeit	++	+	+	+	+
Betrachtung der Datenentstehung	o	o	o	o	o
Behandlung ortsbezogener Daten	o	o	o	o	o
Benutzerschnittstelle	++	o	++	+	+
Domänenunabhängigkeit	+	+	+	+	+

Tabelle 2-4: Vergleich der Leistungsmerkmale verschiedener Ansätze zur inhaltlichen Indizierung und Annotation und Bewertung dieser Anwendungsklasse

wendungsszenarien miteinander verglichen und weiterhin wird eine abschließende Bewertung der betrachteten Anwendungsklasse vorgenommen. Ein Einsatz von semantischen Technologien zur Annotation von Messdaten findet leider nicht statt. Die zu Beginn des Kapitels vorgestellten Methoden und Werkzeuge zur Verarbeitung von Messdaten profitieren somit nicht von diesen neuen Technologien.

Im Rahmen dieser Arbeit soll diese Kombination eingeführt werden. Somit wird es ermöglicht, Messdaten auf feingranularer Ebene mit formalen Methoden zu beschreiben und mit externen Modellen und Systemen zu verknüpfen. Der folgende Abschnitt liefert einen abschließenden Vergleich über die zuvor behandelten Ansätze und legt dar, wie diese sich ergänzen.

2.2.6 Vergleich und Bewertung verwandter Arbeiten

Entsprechend der Forschungsfrage dieser Arbeit sollen Messdaten anhand von Metadaten so beschrieben werden, dass formalisiertes Domänenwissen am Beispiel von Fahrmanövern verfügbar wird. Die verschiedenen Bereiche der verwandten Arbeiten beschäftigen sich alle mit Datenprodukten und in verschiedener Form mit Metadaten, die diese beschreiben. Tabelle 2-5 gibt eine abschließende Übersicht über die Bereiche der diskutierten verwandten Arbeiten und deren Unterstützung verschiedener Leistungsmerkmale basierend auf den Bewertungen der Anwendungsklassen in den Tabellen 2-1 bis 2-4. Im Hinblick auf die Metadaten setzen die Bereiche unterschiedliche Schwerpunkte und betrachten verschiedene Phasen im Lebenszyklus der Datenprodukte. Während die SWS die Datenentstehung fokussieren, betrachten die Informationssysteme die Analyse der Daten und die Datenhaltungssysteme deren Archivierung. Um zu verdeutlichen, dass die inhaltliche Indizierung und Annotation nicht mehr Messdaten, sondern Metadaten und semantische Technologien fokussieren, ist dieser Bereich in der Tabelle optisch abgetrennt.

		Informations-systeme für Fahrversuche	Scientific Workflow Systems	Wissenschaftliche Datenhaltungs-systeme	Inhaltliche Indizierung und Annotation
Einsatzgebiet		Messdaten-visualisierung, Plots, Videos	Modellierung und Ausführung von Workflows	Archivierung von Messdaten	Inhaltliche Beschreibung
Produkte (Beispiele) bzw. Kategorien		Fahrerleistungs-datenbank, Ærogator	B-Fabric, Kepler, Taverna, Triana DIMS, DataFinder	PANGAEA, GBIF, ILTER, MORIS	Web-, Text-, Video-, Audio-Annotation
Leistungsmerkmale	Datenverwaltung	++	+	+	+
	Interpretation der Dateninhalte	++	o	o	++
	Datenschema	+	o	o	o
	Metadaten	o	o	+	++
	Durchsuchbarkeit	+	o	++	+
	Betrachtung der Datenentstehung	o	++	o	o
	Behandlung orts-bezogener Daten	+	o	++	o
	Benutzer-schnittstelle	++	+	+	+
	Domänen-unabhängigkeit	o	++	+	+

Tabelle 2-5: Übersicht der behandelten Kategorien verwandter Arbeiten und Vergleich der jeweiligen Leistungsmerkmale

Bezüglich der spezialisierten Informationssysteme zur Verwaltung von Fahrexperimenten lässt sich festhalten, dass diese besonders benutzerfreundlich sind, um dem Anwender ein intuitives und komfortables Arbeiten mit den Messdaten zu ermöglichen. Sie unterstützen dagegen nicht bei der Verwaltung zusätzlicher Metadaten, einer formalen Beschreibung oder austauschbaren Modellen. Diese Eigenschaften sind jedoch wichtig für die Dokumentation, automatisierte Schlussfolgerungen und eine verbesserte Wiederverwendbarkeit. Mit den in dieser Arbeit vorgestellten Konzepten, lassen sich diese Fähigkeiten ergänzen.

Der Fokus der SWS liegt auf der Kombination der Verarbeitungsschritte und der Datenentstehung. Die Daten selber werden jedoch immer als Blackbox betrachtet und nur mit allgemeinen Metainformationen auf Datei-Ebene versehen. Somit ist es leider nicht möglich, sich auf einzelne Messwerte zu beziehen. Genau an dieser Stelle möchte die vorliegende Arbeit ansetzen und bei den Betrachtungen den Blick auf die Interpretation der Dateninhalte in den Fokus setzen, um so die Auswertung der Daten zusätzlich zu unterstützen.

Die vorgestellten wissenschaftlichen Datenhaltungssysteme sind besonders leicht zu durchsuchen. Sie betrachten die gespeicherten Datenprodukte ebenfalls wie eine Blackbox. Somit lassen sich auf der Daten-Ebene keine formalen Aussagen treffen, sodass komplexe Suchanfragen nach Datenpunkten und automatisierte Schlussfolgerungen auf diesen nicht möglich sind. Durch eine Integration semantischer Technologien, wie sie in dieser Arbeit betrachtet wird, ergeben sich diese neuen Möglichkeiten.

Der Einsatz von Annotationen und semantischen Technologien für die Inhalte und Metadaten von Webseiten und Mediadaten ist dagegen ein etablierter Ansatz. Eine Adaption für Mess-

und Sensordaten ist jedoch leider nicht bekannt. Eine solche soll daher im weiteren Verlauf wesentlicher Gegenstand der vorliegenden Arbeit sein. Weiterhin lässt sich feststellen, dass in den vorgestellten Arbeiten die Fähigkeiten semantischer Technologien, Schlussfolgerungen zu ziehen (Reasoning), kaum eingesetzt werden. Daher wird in dieser Arbeit das automatische Schlussfolgern für Messdaten in der betrachteten Domäne evaluiert, sodass aus dem eingesetzten Reasoning ein Mehrwert generiert wird.

Zusammenfassend lässt sich festhalten, dass keines der betrachteten Systeme verwandter Arbeiten wissenschaftliche Messdaten verarbeitet und auf feingranularer Ebene einzelner Messwerte semantische Technologien zur Beschreibung verwendet. An dieser Stelle setzt die vorliegende Arbeit im weiteren Verlauf an und schließt somit eine Betrachtungslücke, um zusätzliche Möglichkeiten und Vorteile zu erschließen. Im folgenden Abschnitt werden zunächst die Grundlagen von semantischen Technologien eingeführt, um diese dann zur Lösung der aufgezeigten Problemstellung einsetzen zu können.

2.3 Semantische Technologien

Es folgt eine Einführung und Übersicht über die verschiedenen Standards, welche als semantische Technologien bezeichnet werden und die Basis für das Semantic Web bilden. Die Einführung erfolgt in Anlehnung an [HKR08] und die jeweiligen W3C-Standardisierungsdokumente (u.a. [Hay04, MPSP09]).

2.3.1 Resource Description Framework (RDF)

Das *Resource Description Framework* (RDF, [MM04, Mil98]) ist ein System zur Beschreibung von Ressourcen und wird in Anlehnung an [HKR08, Kap. 2 u. 3, S. 17 ff.] eingeführt. Es wurde vom W3C im Jahre 2004 in seiner ersten finalen Version standardisiert. Ursprünglich wurde RDF zur Beschreibung von Metadaten im *World Wide Web* (WWW) entworfen, jedoch ist RDF mittlerweile ein allgemein gebräuchliches Mittel zur Beschreibung von Informationen.

2.3.1.1 Grundlagen

RDF baut auf *Uniform Resource Identifier* (URI, [BLFM05]), der *Extensible Markup Language* (XML, [BPSM$^+$06]) und Namensräumen [BHL$^+$09] auf, die zuerst sehr kurz eingeführt werden. Insbesondere zu XML wird Grundlagenwissen vorausgesetzt.

XML bildet eine syntaktische Grundlage zur Beschreibung von Dokumenten. Mit Tags können Teile eines (Text-)Dokumentes ausgezeichnet werden. Die Interpretation der spezifischen Tags ist

jedoch immer anwendungsabhängig. So stellt z.B. die auf XML aufbauende *Extensible Hypertext Markup Language* (XHTML) ein XML-Tag-Vokabular zur Beschreibung von Webseiten dar. Allgemein betrachtet ist XML eine Grundlage zur Definition von Markup-Sprachen. Die Struktur von XML-Dokumenten entspricht immer logischen Bäumen mit nur einem Wurzelelement.

Eine im Folgenden sehr wichtige Eigenschaft von XML ist die Modularisierbarkeit mittels URIs und Namensräumen. Dabei ist eine URI eine einfache Zeichenfolge, die abstrakte oder physische Ressourcen identifiziert. Eine besonders bekannte Rolle spielen URIs bei der Bezeichnung von Webseiten (z.B. `http://www.dlr.de`). URIs wurden ursprünglich in *Uniform Resource Locators* (URL) und *Uniform Resource Names* (URN) unterteilt, was heute jedoch nicht mehr so gehandhabt wird. Es ist jedoch nicht zwingend notwendig, dass URIs Webdokumente spezifizieren. Vielmehr ist es wichtig, dass ein Konsens besteht, welche reale Ressource durch eine URI repräsentiert wird.

URIs sind hierarchisch aufgebaut, sodass einer hierarchisch wichtigen Ressource mehrere Ressourcen niedriger Hierarchiestufen zugeordnet werden. Mittels Fragmenten, die über ein Hash (#) gekennzeichnet werden, lassen sich Elemente innerhalb von einer URI referenzierten Ressource kennzeichnen.

Aufgrund der hierarchischen Struktur von URIs lassen diese sich relativ zu einer zuvor zu bestimmenden Basis angeben. Durch die Einführung von *Namensräumen* wird ein Mechanismus geschaffen, über den flexibel eine solche Basis durch die Verwendung von Abkürzungen angegeben werden kann. Somit lassen sich URIs bei Bedarf deutlich kompakter darstellen.

XML ist zusammen mit den dazugehörigen Standards eine wichtige Basistechnologie. Daher bildet XML mit die Grundlage für das im Folgenden eingeführte RDF(S) und OWL. Was XML jedoch nicht leisten kann, ist die allgemeine Interpretation der XML-Tags, sodass eine Maschine nicht den Inhalt und die Semantik eines unbekannten XML-Dokumentes bestimmen kann. Beim Interpretieren sind menschliche Anwender im Vorteil, da sie bei sinnvoller Benennung von XML-Tags, deren Bedeutung erraten können.

2.3.1.2 Datenmodell von RDF

Das Ziel von RDF ist der Austausch von Daten und dabei deren Bedeutung zu erhalten. Zu diesem Zweck ist RDF eine Sprache zur Beschreibung von Aussagen und Informationen. Auf Basis der folgenden Definitionen von Graphen und RDF wird zur Klärung der Forschungsfrage im weiteren Verlauf dieser Arbeit Domänenwissen am Beispiel von Fahrmanövern formal dargestellt. Ein RDF-Dokument beschreibt immer einen gerichteten Graphen (Def. nach [CLR01, Kap. 5.4, S. 86 – 90]), der aus Knoten und gerichteten Kanten besteht (im Gegensatz zu XML, das Bäume beschreibt).

Definition 1 ((Gerichteter) Graph) *Ein (gerichteter) Graph G ist ein Tupel (V, E), wobei V die Menge der* Knoten *(englisch* vertex*) und E eine Menge von* Kanten *(englisch* edge*) ist. Dabei ist E in gerichteten Graphen (mit Mehrfachkanten) eine (Multi-)Menge über dem kartesischen Produkt $V \times V$.*

Graphen sind ein allgemeineres mathematisches Darstellungsmittel als Bäume, da sich mit ihnen nicht nur Hierarchien abbilden lassen. Außerdem können Graphen beliebig in Teilgraphen zerteilt oder aus ihnen zusammengesetzt werden, sodass sich mehrere Dokumente flexibel kombinieren lassen. Im Gegensatz zu einfachen, gerichteten Graphen, werden bei RDF nicht nur den Knoten, sondern auch den Kanten Elemente zugeordnet.

Definition 2 (Graph mit bestimmten Kanten) *Für einen gerichteten Graphen mit bestimmten Kanten, bei dem die Elemente zur eindeutigen Benennung der Knoten und Kanten in der Menge V zusammengefasst werden, ist E eine Teilmenge des kartesischen Produktes $V \times V \times V$.*

In RDF werden die Knoten und Kanten der Bäume mit URIs benannt, um sie eindeutig zu kennzeichnen. Die URIs beschreiben abstrakte Ressourcen (z.B. Personen, Orte, Institutionen, Beziehungen ...) und deren Behandlung kann von der jeweiligen Anwendung abhängig sein. Ausnahmen stellen *Blank Nodes* (leere Knoten bzw. BNodes) und Literale dar. Blank Nodes sind anonyme Ressourcen, für die kein expliziter Name vergeben wird. Um sie dennoch handhaben und unterscheiden zu können, werden in Computeranwendungen oder RDF-Serialisierungen i.d.R. lokal eindeutige Bezeichner eingeführt. Logisch betrachtet besitzen sie jedoch keine Beschriftung.

Literale dagegen repräsentieren konkrete Datenwerte (z.B. Zahlen oder Zeichenketten). Darüber hinaus können Literale einen Datentyp besitzen, der deren Interpretation beeinflusst. Die Zeichenketten „1“ und „01“ sind unterschiedlich, können jedoch als Zahl betrachtet identisch sein. Sehr gebräuchlich ist die Verwendung der vom XML-Schema definierten Datentypen (Abschnitt D.11) im Namensraum `xsd` (vgl. [BPM04, CP06]). Weiterhin können Literale eine Sprachangabe besitzen. Jedoch dürfen Literale niemals zugleich eine Sprachangabe und einen Typ besitzen und sie sind niemals Ausgangspunkte für Kanten in einem RDF-Graphen.

Definition 3 (RDF-Modell) *Ein RDF-Modell M ist ein 4-Tupel $(R, P, \mathcal{L}, S)$. Die Menge R wird als Ressourcen, die Menge $P \subseteq R$ als Properties (Eigenschaften), $\mathcal{L} \not\subset R$ als Literale und $S \subseteq R \times P \times (R \cup \mathcal{L})$ als Statements (Aussagen) bezeichnet.*

Die hiermit eingeführten Definitionen bilden die Grundlage für die formale Darstellung von Wissensmodellen, was im weiteren Verlauf der Arbeit anhand von Fahrmanövern demonstriert wird. Da sich Graphen leicht graphisch darstellen lassen, wird oft diese Darstellungsform gewählt. Insbesondere baut Abschnitt 3.4.2 auf diesen Definitionen auf, um eine ansprechende Visualisierung einzuführen.

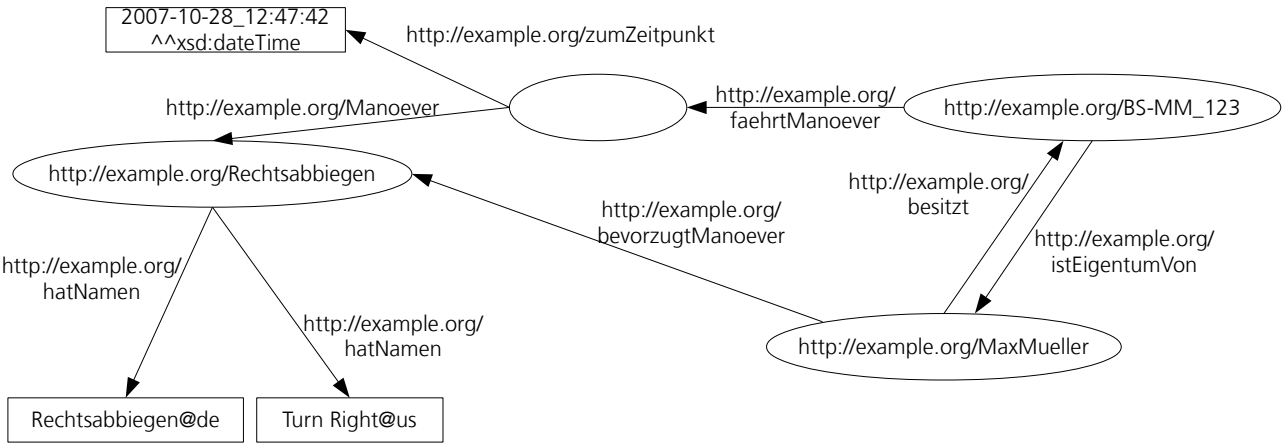


Abbildung 2-11: Beispiel für einen RDF-Graphen

Die durch die Properties gebildeten Kanten werden in der Darstellung als Graph durch Pfeile zwischen den Knoten repräsentiert. Als Beschriftung der Pfeile werden die URIs der jeweiligen Properties verwendet. Ressourcen, die durch eine URI repräsentiert werden, werden als ovale Knoten dargestellt. Das trifft ebenfalls auf Blank Nodes zu. Literale dagegen werden als Rechtecke dargestellt. Ein Beispiel für einen RDF-Graphen über Fahrer, Fahrzeuge und Manöver ist in Abb. 2-11 dargestellt. In der Abbildung ist weiterhin zu sehen, dass Angaben zur Sprache mittels eines @ und Datentypen mittels ^^ gekennzeichnet werden. Die URL des Knotens, der ein konkretes Fahrzeug repräsentieren soll, ist an ein fiktives Kennzeichen angelehnt. Um die Modellierung der Manöverausführung mit Zeitangabe umzusetzen, wird ein Hilfsknoten eingesetzt, der durch einen Blank Node realisiert ist.

2.3.1.3 Syntax von RDF

Zur Verarbeitung in Computern werden visuelle Graphen in ein äquivalentes Datenmodell umgewandelt. Da RDF-Graphen tendenziell lichte Graphen sind, sind Erreichbarkeitsmatrizen keine effiziente Wahl. Daher werden die Kanten des Graphen einzeln als *Tripel* mit Anfangsknoten, Kantenbeschriftung und Endknoten gespeichert. Bei den Elementen dieser RDF-Tripel wird von *Subjekt*, *Prädikat* und *Objekt* bzw. *Subject*, *Predicate* und *Object* gesprochen. Daher stammt die alternative Bezeichnung *RDF Statement* (Aussage) für Tripel.

Bei einer Speicherung eines Graphen als Tripel ist die Reihenfolge der Tripel nicht von Bedeutung. Ein erster Vorschlag von Berners-Lee für eine serialisierte Darstellung heißt *Notation 3* (N3, [BL06b]). Im Rahmen der RDF-Recommendation [Hay04] wird eine vereinfachte Version von N3 mit dem Namen *N-Triples* verwendet. Eine Erweiterung von N-Triples um Kurzschreibweisen heißt *Turtle*. Zur Unterscheidung werden URIs in spitzen Klammern und Literale in Anführungszeichen geschrieben. Ein Punkt beendet vollständige Aussagen in Form eines Tripels. Jedoch können in der Kurzschreibweise mehrere Aussagen mit dem gleichen Subjekt zusammengefasst werden, was durch ein Semikolon gekennzeichnet wird. Zur kompakteren Schreibweise

```
@prefix ex:       <http://example.org/> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .

ex:MaxMueller
       ex:besitzt ex:BS-MM_123 .

ex:BS-MM_123
       ex:faehrtManoever
               [ ex:Manoever ex:Rechtsabbiegen ;
                 ex:zumZeitpunkt "2007-10-28_12:47:42"^^xsd:dateTime
               ] .
```

Listing 2-1: Darstellung eines RDF-Graphen in Turtle

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://example.org/" >
  <rdf:Description rdf:about="MaxMueller">
    <ex:besitzt rdf:resource="BS-MM_123"/>
  </rdf:Description>
  <rdf:Description rdf:about="BS-MM_123">
    <ex:faehrtManoever rdf:nodeID="A0"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <ex:Manoever rdf:resource="Rechtsabbiegen"/>
    <ex:zumZeitpunkt rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
      2007-10-28T12:47:42Z
    </ex:zumZeitpunkt>
  </rdf:Description>
</rdf:RDF>
```

Listing 2-2: Darstellung eines RDF-Graphen in RDF/XML

werden XML-Namensräume unterstützt. Listing 2-1 zeigt einen Teilgraphen von Abb. 2-11 in Turtle-Schreibweise.

Turtle ist sowohl für Menschen gut verständlich als auch durch Maschinen leicht verarbeitbar. Als gleichwertige Alternative existiert ein auf XML basierendes Austauschformat mit dem Namen *RDF/XML* [Bec04], welches für menschliche Leser jedoch oft nicht ganz so intuitiv lesbar ist wie Turtle. Dass XML nur Bäume darstellt, ist für die Darstellung von RDF kein Hindernis, da ebenfalls wieder eine auf Tripeln basierende Serialisierung erfolgt. Für die XML/RDF-Serialisierung werden diverse XML-Tags mit dem Namensraum-Präfix `rdf` verwendet. Das Subjekt eines Tripels wird mit dem `about`-Attribut in einem `Description`-Element gekennzeichnet. Das Prädikat wird direkt als Kindelement des umschließenden Subjekt-Elementes abgebildet. Außerdem werden für Blank Nodes explizit `nodeID`-Kennzeichner eingeführt, um unterschiedliche Blank Nodes unterscheiden zu können. Das Beispiel aus Listing 2-1 ist als äquivalente Version in XML/RDF in Listing 2-2 dargestellt.

Zur Darstellung komplexer Zusammenhänge existieren in RDF weitere komplexe Ausdrucksmittel. Um in RDF zu verdeutlichen, dass URIs insbesondere als Prädikat verwendet werden sollen, können diese als *Property* (Eigenschaft) deklariert werden. RDF-Tripel erlauben eigentlich nur binäre Beziehungen zwischen Knoten. Um Mehrfachbeziehungen zu beschreiben, können Hilfsknoten, z.B. in Form von Blank Nodes, verwendet werden. Ein besonderes Beispiel für mehrwertige Beziehungen stellt die *Reifikation* (Vergegenständlichung oder Verdinglichung) dar. Bei

der Reifikation werden Aussagen über andere Aussagen (Tripel) getroffen. Weitere Ausdrucksmittel, welche mit Hilfsknoten arbeiten, sind verschiedene Formen von Listen und Mengen.

Für eine semantisch eindeutige Interpretation der in RDF codierten Informationen ist es weiterhin notwendig, dass wohldefinierte *Vokabulare* verwendet werden. Darunter sind Zusammenstellungen von Bezeichnern mit klar definierter Bedeutung zu verstehen. Existiert für den gewünschten Anwendungsfall kein Vokabular, so muss dies bei Bedarf neu definiert werden. Bekannte Beispiele für etablierte Vokabulare sind *Friend of a Friend* (FOAF, [Bri10, DZFJ05]) und elektronische Visitenkarten (vCards) in RDF [HISW10]. Auf dieser Basis können Computer bereits Aussagen treffen, deren Interpretation obliegt jedoch weiterhin dem menschlichen Anwender. Um diesen Prozess weiter zu unterstützen, werden im Anschluss an den nächsten Abschnitt *RDF Schema* und *OWL* eingeführt.

2.3.1.4 Formale Semantik

In diesem Abschnitt wird die *logische Dimension* des Semantik-Begriffs für RDF nach [HKR08, Kap. 4, S. 91 ff.] skizziert, die als *formale Semantik* bezeichnet wird und offiziell in [Hay04] definiert ist. Das Ziel formaler Logik ist das korrekte und einheitliche Schlussfolgern auf vorliegenden Graphen. Aussagen in der Logik heißen *Sätze* und entsprechen den RDF-Tripeln.

Grundlegende Idee einer modelltheoretischen Semantik ist es, Aussagen über die reale Welt zu treffen. Elemente des betrachteten Modells repräsentieren reale Ressourcen und abstrahieren somit die Realität. Gegenstand der formalen Semantik ist die Definition für die Interpretation des Modells. Zu diesem Zweck werden Vereinbarungen getroffen, wie Aussagen in einem betrachteten Modell zu interpretieren sind.

Zunächst wird in der Literatur das formale Konstrukt einer *einfachen Interpretation* I auf einem Vokabular $\mathcal{V}$ eingeführt (Abschnitt D.1). Die Interpretation umfasst eine nichtleere Menge von *Ressourcen* R, von *Properties* P und von *ungetypten Literalen* $LV \subset R$. Weiterhin umfasst I drei Funktionen $\mathrm{I_{EXT}}$, $\mathrm{I_S}$ und $\mathrm{I_L}$, welche die Mengen zueinander in Beziehung setzen. $\mathrm{I_{EXT}}$ beschreibt diejenigen Tripel, welche Aussagen in dem betrachteten Modell sind. $\mathrm{I_S}$ bildet Namen aus $\mathcal{V}$, welche URIs sind, auf R und P ab. Für I wird weiterhin eine Interpretationsfunktion $\cdot^{\mathcal{I}}$ definiert, welche die soeben angegebenen Aussagen umfasst und zusätzlich Sprach- und Typangaben für Literale mit Hilfe von $\mathrm{I_L}$ behandelt.

Jedem *grundierten* Tripel (ohne leere Knoten) weist die Interpretationsfunktion $\cdot^{\mathcal{I}}$ einen Wahrheitswert (*wahr* oder *falsch*) zu. Der Wahrheitswert $s\ p\ o.^{\mathcal{I}}$, der einem grundierten Tripel $s\ p\ o.$ zugeordnet wird, ist genau dann *wahr*, wenn die Bestandteile $s, p, o \in \mathcal{V}$ und $\langle s^{\mathcal{I}}, o^{\mathcal{I}} \rangle \in \mathrm{I_{EXT}}(p^{\mathcal{I}})$.

Basierend auf den Wahrheitswerten für grundierte Tripel T erfolgt die Definition von $\cdot^{\mathcal{I}}$, die jedem grundierten Graphen G ebenfalls einen Wahrheitswert zuweist. Es gilt $G^{\mathcal{I}} = wahr$ genau

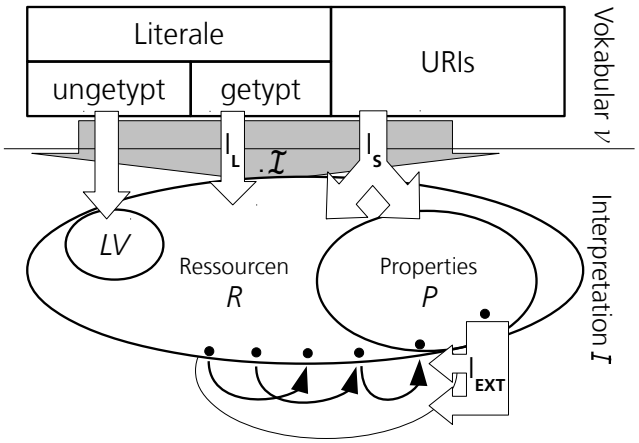


Abbildung 2-12: Schematische Darstellung einer RDF-Interpretation[2] (nach [HKR08, S. 95])

dann, wenn $T^{\mathcal{I}} = wahr$ für alle Tripel $T \in G$. Die Erweiterung von grundierten Graphen um leere Knoten erfolgt in [HKR08, Hay04].

Die bisher betrachtete einfache Interpretation behandelt alle URIs im Vokabular $\mathcal{V}$ gleich. Eine *RDF-Interpretation* (Abb. 2-12, Abschnitt D.1) erweitert diese um eine spezielle Behandlung für das RDF-Vokabular $\mathcal{V}_{RDF}$. Die RDF-Interpretation sieht vor, dass eine URI $x \in P$ genau dann, wenn $\langle x, \texttt{rdf:Property}^{\mathcal{I}} \rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$. Weiterhin wird von Literalen vom Typ `rdf:XMLLiteral` gefordert, dass sie wohlgeformte XML-Ausdrücke darstellen.

Zusätzlich werden bestimmte *axiomatische RDF-Tripel*, die in Abschnitt D.2 aufgeführt sind, immer als wahr ausgewertet. Ihre Aufgabe ist weitgehend die Kennzeichnung spezieller Ressourcen als Property.

2.3.2 RDF Schema (RDFS)

In der bisher eingeführten Form umfasst RDF einzelne Individuen bzw. Ressourcen, deren Beziehungen und Literale. Im Folgenden werden diese Sprachmittel nach [HKR08, Kap. 3 u. 4, S. 66 ff.] und [Hay04] durch *RDF Schema* (RDFS) um eine Ebene für *terminologisches Wissen* bzw. *Schemawissen* erweitert. Unter Schemawissen ist allgemeingültiges Wissen zu verstehen, welches für Individuen vom gleichen Typ gültig ist. Aussagen bezüglich einzelner Individuen dagegen werden als *assertionales Instanzwissen* bezeichnet. Abbildung 2-13 zeigt ein entsprechendes Beispiel in Anlehnung an Abb. 2-11 aus dem Bereich der Fahrmanöver.

RDFS wird aufgrund der Möglichkeit, Schemawissen zu beschreiben, bereits als *Wissensrepräsentations-* oder *Ontologie-Sprache* bezeichnet. Unter einer Ontologie wird in der Informatik eine formale Darstellung von Begriffen und ihrer Beziehungen zueinander zur Modellierung des Wissens einer bestimmten Domäne verstanden (vgl. [Lz09, UG96]). Eine Ontologie kann somit als eine formale Darstellung einer Wissensbasis (Knowledge Base, KB) betrachtet werden (vgl. [GG95]). Zu dem Zweck, Schemawissen darzustellen, wird zunächst ein

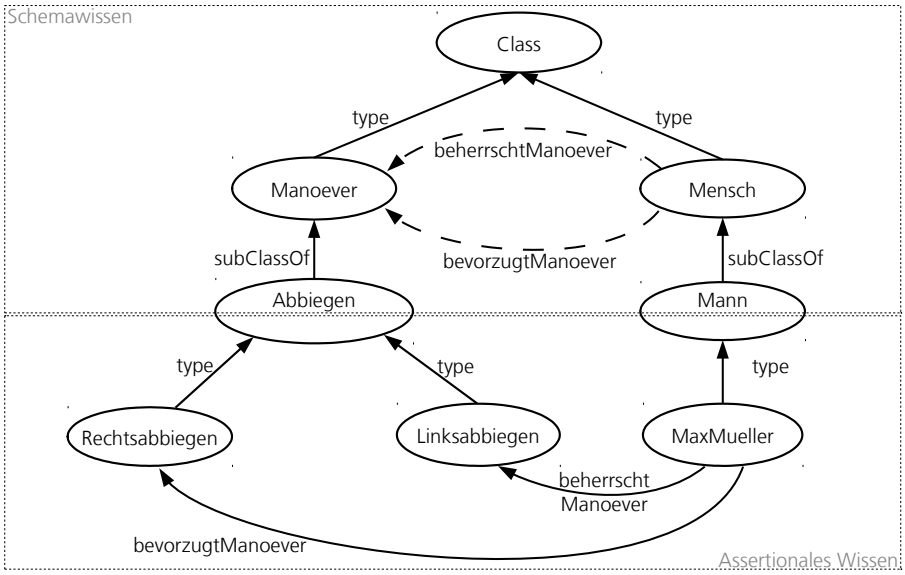


Abbildung 2-13: Beispiel für einen RDFS-Graphen in Anlehnung an Abb. 2-11

```
1  ex:Mensch
2    rdf:type rdfs:Class .
3
4  ex:Manoever
5    rdf:type owl:Class .
6
7  ex:Abbiegen
8    rdf:type rdfs:Class ;
9    rdfs:subClassOf ex:Manoever .
10
11 ex:beherrschtManoever
12   rdf:type rdf:Property ;
13   rdfs:domain ex:Mensch ;
14   rdfs:range ex:Manoever .
15
16 ex:bevorzugtManoever
17   rdfs:subPropertyOf ex:beherrschtManoever .
```

Listing 2-3: Deklaration der Klassen `Mensch`, `Abbiegen` und `Manoever` und deren Beziehungen in RDFS in Anlehnung an Abb. 2-13

zusätzliches Vokabular in einem separaten Namensraum `rdfs` eingeführt, das als Erweiterung zum regulären RDF-Vokabular fungiert.

2.3.2.1 Klassen, Instanzen und Properties

RDFS führt ähnlich der objektorientierten Programmierung das Prinzip von Klassen und Instanzen einer Klasse ein. Hierbei beschreibt eine Klasse eine Menge von Ressourcen mit gemeinsamen Eigenschaften. URIs, die eine Klasse repräsentieren, heißen somit *Klassenbezeichner*. Die Abb. 2-13 zeigt in der oberen Hälfte Klassen und deren Beziehungen und in der unteren Hälfte die Instanzen. Die Klassen `Mensch`, `Manoever` und `Abbiegen` und deren Beziehungen werden wie in Listing 2-3 deklariert.

Außerdem gibt es in RDFS einige vordefinierte Klassenbezeichner mit fester Bedeutung (z.B. Resource, Property, Statement). Mittels der transitiven und reflexiven Relation `rdfs:subClassOf`

werden Hierarchien von Klassen beschrieben. In dem Beispiel ist somit `Mann` eine Unterklasse von `Mensch`. Mit diesen Sprachmitteln lassen sich bereits Klassensysteme, auch *Taxonomien* [BW00] genannt, wie sie z.B. in der Systematik [LG01] in der Biologie verwendet werden, beschreiben.

In RDFS wird das bereits in RDF eingeführte Konzept der Properties weiter verfeinert, sodass es für diese möglich ist, Hierarchien zu definieren. So lässt sich im Beispiel mittels `rdfs:subPropertyOf` die Eigenschaft `bevorzugtManoever` als Sub-Property von `beherrschtManoever` realisieren. Weiterhin erlaubt RDFS die Einschränkung von Properties auf sinnvolle Subjekte und Objekte durch Klassenrestriktionen. Mittels `rdfs:domain` wird der Definitionsbereich von Properties auf ausgewählte Klassen eingeschränkt. Zum Einschränken des Wertebereichs wird `rdfs:range` verwendet. In Abb. 2-13 sind diese Umstände durch gestrichelte Pfeile dargestellt.

2.3.2.2 Zusätzliche Ausdrucksmittel

Über die bisher beschriebenen Ausdrucksmittel hinaus bietet RDFS noch einige weitere nützliche Sprachkonstrukte. Mit dem Property `rdfs:label` lassen sich alle Ressourcen mit einem menschenlesbaren Literal als Beschriftung versehen. Das Property `rdfs:comment` erlaubt analog die Verknüpfung mit einem für Menschen verständlichen Kommentar. Zum Verweis auf verwandte Ressourcen dienen weiterhin die Eigenschaften `rdfs:seeAlso` und `rdfs:isDefinedBy`.

Es ist zu erwähnen, dass bei der Benennung von URIs im Allgemeinen mit der Konvention gearbeitet wird, dass Klassen mit einem Großbuchstaben und URIs für Instanzen und Properties mit einem kleinen Buchstaben beginnen. Eine vollständige Übersicht über die Sprachelemente von RDF(S) ist in Abschnitt D.5 aufgeführt.

2.3.2.3 Formale Semantik

Für die formale Semantik von RDFS wird die in Abschnitt 2.3.1.4 eingeführte RDF-Interpretation zu einer *RDFS-Interpretation* nach [HKR08, Kap. 4, S. 91 ff.] bzw. [Hay04] erweitert (Abschnitt D.3), welche auf dem RDFS-Vokabular $\mathcal{V}_{RDFS}$ arbeitet.

Zunächst wird eine neue Funktion $\mathrm{I}_{\mathrm{CEXT}}\colon R \to 2^R$ eingeführt, welche einer Klasse die darin enthaltenen Ressourcen zuweist. Mit deren Hilfe werden die Mengen $R = \mathrm{I}_{\mathrm{CEXT}}(\texttt{rdfs:Resource}^{\mathcal{I}})$ und $LV = \mathrm{I}_{\mathrm{CEXT}}(\texttt{rdfs:Literal}^{\mathcal{I}})$ definiert und die neue Menge der Klassen $C = \mathrm{I}_{\mathrm{CEXT}}(\texttt{rdfs:Class}^{\mathcal{I}})$ eingeführt.

Als weitere Bedingungen werden für RDFS definiert, dass wenn ein Property x eine Klasse y als Domain / Range besitzt und eine Ressource u als Subjekt / Objekt mit x benutzt wird, dann ist u vom Typ der Klasse y. Weiterhin werden die `rdfs:subPropertyOf-` und `rdfs:subClassOf-`Eigenschaften auf den entsprechenden Mengen als reflexiv und transitiv definiert. Diese Bedingungen müssen mindestens erfüllt sein und werden *intensionale Semantik* genannt.

Eine strengere Auslegung der RDFS-Interpretation fordert zudem, dass für eine Unterklasse / ein Sub-Property gilt, dass sie *echt* enthalten ist in der Klasse / Property, die sie spezialisiert. Weiterhin kann gefordert werden, dass die Bedingungen für die Domain- und Range-Angaben nicht über eine Implikation, sondern über eine Äquivalenz verknüpft sind, sodass `rdfs:domain`- und `rdfs:range`-Aussagen automatisiert geschlussfolgert werden könnten. Diese zusätzlichen Bedingungen ergeben die sogenannte *extensionale Semantik* von RDFS, sind aber deutlich schwieriger in Werkzeugen zu implementieren und daher nur selten anzutreffen. Weiterhin muss eine RDFS-Interpretation die in Abschnitt D.4 aufgeführten axiomatischen Tripel immer erfüllen.

2.3.2.4 Grenzen

Anhand der in RDFS eingeführten Sprachkonstrukte lassen sich bereits sinnvolle Ontologien mit Klassensystemen und deren Eigenschaften und Zusammenhängen untereinander beschreiben. Durch Einsatz der in Abschnitt 2.3.1.4 und 2.3.2.3 eingeführten Interpretationen, können durch Schlussfolgerungen bereits wertvolle Aussagen über die modellierten Ontologien getroffen werden. Beispielsweise lassen sich Inkonsistenzen beim Einsatz von Properties erkennen, Abfragen durchführen, ob Individuen/Klassen zu einer bestimmten anderen Klasse gehören, oder alle Individuen einer konkreten Klasse aufzählen.

RDFS baut jedoch auf der *Open World Assumption* auf, dass nur über das im aktuell betrachteten Graphen vorhandene Wissen Aussagen getroffen werden können und außerhalb noch mehr nicht betrachtetes Wissen existiert, über das keine Aussagen getroffen werden können. Daher kann in RDFS grundsätzlich nicht dargestellt werden, dass Aussagen nicht wahr sind, da sie einfach nur nicht Bestandteil des betrachteten Graphen sein könnten. Was durchaus möglich ist, ist die explizite Modellierung von Negationen. So lassen sich in einer fiktiven Ontologie durchaus die Klassen *Neuwagen* und *Gebrauchtwagen* modellieren. Es ist jedoch in RDFS nicht möglich auszudrücken, dass der Durchschnitt dieser zwei Mengen die leere Menge ist. (vgl. [HKR08, Kap. 4.4, S. 119])

2.3.3 Web Ontology Language (OWL)

Das zuvor eingeführte RDFS ist bereits in der Lage, einfache Ontologien zu modellieren und daraus bei guter Laufzeit implizites Wissen abzuleiten. Aufgabe der *Web Ontology Language* (OWL) ist es, diese Möglichkeiten zu erweitern, mit dem Ziel formale Logik zur Beschreibung einsetzen zu können (vgl. [SWM04]), was zwangsweise zu einer höheren Laufzeitkomplexität führt. OWL wurde 2004 vom W3C als Standard spezifiziert (vgl. [MvH04, BvHH+04, HPSvH03]) und wird in Anlehnung an [HKR08, Kap. 5 u. 6, S. 125 ff.] eingeführt. Da das behandelte Forschungsgebiet noch sehr aktuell ist, wird OWL ständig weiterentwickelt. Während der Entstehung dieser Arbeit ist im Oktober 2009 der nachfolgende Standard OWL 2 [W3C09] ent-

standen, wobei jede OWL (1) Ontologie eine gültige OWL 2 Ontologie ist, jedoch in OWL 2 einige neue Fähigkeiten eingeführt werden.

2.3.3.1 Grundlagen

Ziel bei der Entwicklung von OWL war eine ausgewogene Balance zwischen der Ausdrucksstärke zum einen und der Effizienz und Skalierbarkeit zum anderen. Aus diesem Grund sind drei Teilsprachen entstanden *OWL Lite* $\subset$ *OWL DL* $\subset$ *OWL Full*, sodass der Anwender die Wahl zwischen einer sehr guten Laufzeit oder einer hohen Ausdrucksstärke besitzt. Abschnitt D.6 gibt eine Übersicht über deren Eigenschaften. Aus diesen ist zu erkennen, dass OWL DL einen zweckmäßigen Kompromiss darstellt und von Werkzeugen gut unterstützt wird, weswegen es in der Praxis die höchste Bedeutung besitzt.

Zur Serialisierung kann jedes OWL-Dokument auf RDF-Tripel zurückgeführt und als gültiges RDF/XML-Dokument (vgl. Abschnitt 2.3.1.3) dargestellt werden. Darüber hinaus existiert eine *abstrakte OWL-Syntax* ([PSHH04, Kap. 2], in OWL 2 *Manchester-Syntax* [HPS09] genannt). Sie basiert auf einer Syntax für die *Erweiterte Backus-Naur-Form* (EBNF, [ISO96]) und ist für menschliche Anwender leichter lesbar, jedoch nur wenig verbreitet.

Für OWL spezifische Sprachkonstrukte wird analog zu RDF(S) ein neuer Namensraum eingeführt, der üblicherweise mit `owl` abgekürzt wird. OWL-Dokumente tragen einige Informationen, die das Dokument selber beschreiben und in RDF/XML im Dokumentkopf gespeichert werden. Zu diesen Informationen gehören ein Kommentar, umfangreiche Versionsinformationen oder andere Dokumente mit Ontologien, die von dem vorliegenden referenziert werden. Inhalte so importierter Ontologien werden als Teil der vorliegenden Ontologie betrachtet, was die praktische Wiederverwendbarkeit von Ontologien deutlich verbessert.

2.3.3.2 Zusätzliche Ausdrucksmittel

Die Grundelemente von OWL sind wie in RDFS Klassen, Individuen und Properties, wobei letztere in OWL als *Rollen* bezeichnet werden können. Im Unterschied zu RDFS werden die Klassen mit `owl:Class` gekennzeichnet, da sie eine leicht abweichende Interpretation besitzen, und `owl:Class` ist als Unterklasse von `rdfs:Class` definiert. Die Properties werden in OWL strikt in abstrakte *Object-Properties* und in konkrete *Datatype-Properties* unterschieden. Object-Properties verbinden Individuen mit Individuen, während die Datatype-Properties Individuen mit Literalen verknüpfen.

Als Neuerung bei den Klassenbeziehungen ist zu erwähnen, dass jede Klasse als eine Unterklasse von `owl:Thing` interpretiert wird. Weiterhin repräsentiert die Klasse `owl:Nothing` die leere Menge und ist eine Unterklasse aller anderen Klassen. Über die Beziehung `owl:disjointWith` können Klassen außerdem als disjunkt gekennzeichnet werden. Dagegen

dient `owl:equivalentClass` dazu, zwei Klassen als äquivalent zu kennzeichnen, was insbesondere bei der Verknüpfung unterschiedlicher Vokabulare mit überlappendem Wortschatz sinnvoll sein kann. Ähnlich wie für Klassen lassen sich mittels `owl:sameAs` und `owl:differentFrom` Individuen als identisch bzw. als unterschiedlich kennzeichnen. Des Weiteren lassen sich abgeschlossene Klassen als eine feste Menge von Individuen definieren, was im Hinblick auf die Open World Assumption interessant ist, um negative Anfragen formulieren zu können.

Als wichtigste Erweiterung im Vergleich zu RDFS führt OWL *logische Konstruktoren* für Klassen ein. Bisher betrachtete einfache Klassen werden als *atomare* Klassen betrachtet. Mittels logischer Operatoren *und*, *oder* und *nicht*, welche durch die Sprachelemente `owl:intersectionOf`, `owl:unionOf` und `owl:complementOf` abgebildet werden, können atomare Klassen zu *komplexen Klassen* kombiniert werden.

Eine weitere neue Möglichkeit, Klassen zu definieren, ergibt sich aus den *Rolleneinschränkungen*. Rolleneinschränkungen fordern, dass die Individuen, die zu der beschriebenen Klasse gehören, eine bestimmte Property/Rolle mit vorgegebenem Wertebereich und in spezifizierter Anzahl besitzen. So kann beispielsweise eine Klasse *Fahrzeug* definiert werden als etwas, das mit genau vier anderen Individuen der Klasse *Reifen* über das Property *besitztKomponente* verknüpft ist. Die Kardinalität kann auf eine minimale (`owl:minCardinality`), maximale (`owl:maxCardinality`) oder genau vorgegebene (`owl:cardinality`) Größe beschränkt werden. Die Beschränkung des konkreten Wertebereiches erfolgt über `owl:someValuesFrom` und `owl:allValuesFrom`, welche die Funktion von den aus der mathematischen Logik bekannten Quantoren $\exists$ und $\forall$ übernehmen.

Für Properties (bzw. Rollen) existieren zusätzlich zu den bekannten einige neue und ausdrucksstärkere Beschreibungselemente. So lassen sich mit `owl:equivalentProperty` zwei Properties als gleichwertig beschreiben. Mit der Rollenbeziehung `owl:inverseOf` werden dagegen zwei Properties als zueinander *invers* markiert. Eine entsprechende Ontologie vorausgesetzt, ließe sich aus der Aussage `MaxMueller besitzt BS-MM_123` die neue, inverse Aussage `BS-MM_123 istEigentumVon MaxMueller` ableiten. Weiterhin können Properties mit den mathematischen Eigenschaften *transitiv*, *symmetrisch*, *funktional* oder *invers funktional* versehen werden, welche in Abschnitt D.7 formal beschrieben werden. Eine nach Kategorien sortierte Auflistung der für OWL spezifischen Sprachelemente ist in Abschnitt D.8 aufgeführt.

2.3.3.3 Varianten von OWL

Die einzige der drei OWL-Sprachvarianten OWL Full, OWL DL und OWL Lite, die den vollständigen Sprachumfang erlaubt, ist OWL Full. Sie ist die einzige Variante, die den vollständigen Wortschatz von RDFS bereitstellt. Dieser hohe Freiheitsgrad führt zu Herausforderungen bei der Verarbeitung von OWL Full, weswegen OWL DL und OWL Lite entstanden sind. Ursache ist insbesondere, dass OWL Full keine Typentrennung zwischen Individuen, Klassen und Properties/Rollen fordert. Somit sind in OWL Full Aussagen über Klassen von Klassen möglich. Werden

Klassen oder Properties/Rollen an Stelle von Individuen verwendet, um Aussagen über diese zu treffen, wird dies *Metamodellierung* genannt. Diese Möglichkeit führt jedoch dazu, dass OWL Full nicht mehr *entscheidbar* [TW94, Kap. 4, S. 133 ff.] ist und somit eine automatisierte Inferenz nicht mehr uneingeschränkt möglich ist.

OWL DL ist so konzipiert, dass sie einen möglichst großen Sprachumfang von OWL Full erlaubt, jedoch ist sie so eingeschränkt, dass sie entscheidbar ist. Aus dieser Einschränkung ergibt sich insbesondere die Forderung, dass die Beschreibung von Klassen und Instanzen getrennt wird und somit keine Metamodellierung mehr möglich ist. Eine ausführliche Übersicht der Einschränkungen von OWL DL ist in Abschnitt D.9 aufgeführt. Aus diesen Gründen existiert für OWL DL die beste Werkzeugunterstützung und OWL DL spielt in der Praxis und für die vorliegende Arbeit eine besonders herausragende Rolle.

Primäres Entwicklungsziel von OWL Lite war eine möglichst einfache Umsetzbarkeit für Softwarewerkzeuge, weswegen auf noch mehr Sprachelemente als bei OWL DL verzichtet wird. So wird auf die logischen Klassenkonstruktoren verzichtet, was aber dazu führt, dass OWL Lite nicht sehr häufig verwendet wird. Eine Übersicht über die Einschränkungen von OWL Lite ist in Abschnitt D.10 aufgeführt.

2.3.3.4 Formale Semantik

Die formale Semantik von OWL kann analog zu RDF(S) explizit formuliert werden. Eine andere, äquivalente Möglichkeit besteht darin, OWL auf die *Prädikatenlogik erster Stufe* [And86, Kap. 2, S. 45 ff.] zurückzuführen, welche sehr verbreitet und bereits gut erforscht ist. Diese Betrachtung wird im Folgenden nach [HKR08, Kap. 6, S. 163 ff.] ausschnittsweise für wesentliche Sprachelemente von OWL DL durchgeführt, da so der formale Charakter und die Fähigkeiten von OWL gut sichtbar werden.

Historisch gesehen beruht OWL DL auf den Grundlagen *semantischer Netzwerke*. Aus den semantischen Netzwerken sind im Laufe der Zeit formal definierte *Beschreibungslogiken* entstanden, welche ein Fragment der Prädikatenlogik darstellen. Aus diesem Zusammenhang stammt der Name von OWL DL (Description Logic).

Die elementaren Bausteine von OWL sind Klassen, Rollen und Individuen. In der Prädikatenlogik erster Stufe werden Individuen durch Konstantensymbole repräsentiert. Klassen werden auf einstellige und Properties/Rollen auf zweistellige *Prädikate* abgebildet:

$$\mathrm{Abbiegen}(\mathrm{Rechtsabbiegen})$$
$$\mathrm{bevorzugtManoever}(\mathrm{MaxMueller}, \mathrm{Rechtsabbiegen})$$

Der Umstand, dass `Abbiegen` eine Unterklasse (`rdfs:subClassOf`) von `Manoever` ist, wird über eine *Implikation* in der Prädikatenlogik dargestellt:

$$(\forall x)(\text{Abbiegen}(x) \rightarrow \text{Manoever}(x))$$

Die Äquivalenz (`owl:equivalentClass`) der Klassen `Abbiegen` und `Einbiegen` wird in der Prädikatenlogik durch eine *Äquivalenz* abgebildet:

$$(\forall x)(\text{Abbiegen}(x) \leftrightarrow \text{Einbiegen}(x))$$

In Abschnitt 2.3.3.2 wurden *logische Klassenkonstruktoren* eingeführt, wobei *komplexe* Klassen basierend auf *atomaren* durch die Verknüpfung *und*, *oder* und *nicht* definiert werden können. Damit setzt im folgenden Beispiel die Definition der Klasse `Abbiegen` mindestens zwei der drei Bedingungen `Blinken`, `Schulterblick` oder ¬`Geradeausfahren` voraus (vgl. Listing D-1):

$$\begin{aligned}(\forall x)(\text{Abbiegen}(x) \rightarrow &((\text{Blinken}(x) \wedge \text{Schulterblick}(x)) \\ &\vee (\text{Blinken}(x) \wedge \neg\, \text{Geradeausfahren}(x)) \\ &\vee (\text{Schulterblick}(x) \wedge \neg\, \text{Geradeausfahren}(x))))\end{aligned}$$

Weiterhin wurden zuvor OWL-Klassendefinitionen über *Rollenrestriktionen* behandelt. In der Prädikatenlogik werden diese mit Hilfe von Quantoren abgebildet. So wird im folgenden Beispiel alles, was *mindestens* einmal über die Rolle `wirdAusgefuehrtDurch` mit einem `Fahrzeug` verbunden ist, als `Manoever` definiert:

$$(\forall x)(\text{Manoever}(x) \rightarrow (\exists y)(\text{wirdAusgefuehrtDurch}(x, y) \wedge \text{Fahrzeug}(y)))$$

Bei diesem Beispiel ist zu beachten, dass im Vergleich zum vorherigen zwei Quantoren zur Anwendung kommen. Diese Vorgehensweise entspricht der Verwendung des Sprachelementes `owl:someValuesFrom` in OWL. Bei der Übersetzung von `owl:allValuesFrom` wird statt des ∃-Quantors ein ∀-Quantor verwendet.

Mit den vorgestellten Sprachmitteln und deren Umsetzung in Prädikatenlogik lässt sich eine gute Vorstellung von der OWL-Semantik gewinnen. Die Verwendung weiterer Sprachmittel wird in der Literatur ausführlich diskutiert (vgl. [HKR08, PSHH04]).

Grundgedanke von OWL ist die Abbildbarkeit der beschriebenen Wissensbasis auf formale Beschreibungslogiken (vgl. [BHS09, FU10, SH10]). Beschreibungslogiken stellen eine entscheidbare Untermenge der Prädikatenlogik erster Stufe dar, sodass auf dieser Basis automatisiert Schlussfolgerungen berechnet werden können. Die verschiedenen Varianten von OWL bilden verschiedene Beschreibungslogiken ab (vgl. Abschnitt 2.3.3.3, [GHM+09]). OWL Lite entspricht der Beschreibungslogik $\mathcal{SHIF}^{(\mathcal{D})}$. OWL DL bildet $\mathcal{SHOIN}^{(\mathcal{D})}$ ab, während OWL DL aus OWL 2 die mächtigere $\mathcal{SROIQ}^{(\mathcal{D})}$ beschreibt. Die Buchstaben im Namen einer Beschreibungslogik geben an, welche Fähigkeiten die Beschreibungslogik besitzt (vgl. [HKR08, Kap. 6.1.3, S. 172 f.]).

Da OWL Full nicht entscheidbar ist, kann es nicht auf eine Beschreibungslogik abgebildet werden.

Erwähnenswert ist außerdem, dass es in Beschreibungslogiken üblich ist, die getroffenen Aussagen in die *TBox* und die *ABox* einzuteilen. Die TBox enthält das *terminologische* Schemawissen, während die ABox das *assertionales* Instanzwissen enthält (vgl. Abschnitt 2.3.2).

2.3.4 Semantic Web Rule Language (SWRL)

Die *Semantic Web Rule Language* (SWRL, [HPSB+04]) kann als eine Erweiterung von OWL betrachtet werden (vgl. [HP09, MSS05]). SWRL erlaubt es, Regeln als Erweiterung für Ontologien zu definieren. Diese Regeln werden mittels OWL definiert und sind Teil der jeweiligen Ontologie. Aus diesem Grund werden sie von einem Großteil der OWL-Werkzeuge direkt mit unterstützt. Im Rahmen dieser Arbeit wird SWRL dazu benötigt, um Fahrmanöver zu modellieren.

Eine SWRL-Regel besteht immer aus einem *Bedingungsteil* (antecedent bzw. body) und einer *Konsequenz* (consequent bzw. head). Ist der Bedingungsteil erfüllt, dann wird die Konsequenz in dem betrachteten Wissensmodell ebenfalls als erfüllt betrachtet. Insbesondere ist SWRL so konstruiert, dass die Regeln *entscheidbar* sind, sodass sich ein Reasoning (vgl. Abschnitt 2.3.6.2) anwenden lässt (vgl.a. [HP09]).

Sowohl der Term des Bedingungsteils als auch der Term der Konsequenz werden durch Atome gebildet, welche jeweils durch ein logisches *Und* miteinander verknüpft werden. Jedes Atom prüft, ob zwei Elemente der betrachteten Ontologie miteinander über eine Relation verknüpft sind. Für die Darstellung von Klassenzugehörigkeiten gibt es eine abkürzende Darstellung. Über Variablen können Atome verknüpft werden, sodass mehrere Bedingungen für ein Element geprüft werden können oder dieses in der Konsequenz wiederverwendet werden kann. Da die logischen Zusammenhänge als Tripel ausgedrückt schnell sehr umfangreich und komplex werden, gibt es eine besser lesbare Darstellung. Diese Darstellung wird *Human Readable Syntax* [HPSB+04, Kap. 2.2] genannt und stellt logische Zusammenhänge direkt mittels einer entsprechenden mathematischen Notation dar (vgl. Abschnitt 3.5.2).

2.3.5 SPARQL Protocol and RDF Query Language (SPARQL)

Für den praktischen Einsatz von RDF(S) ist es von essentieller Bedeutung, flexibel und zielgerichtet auf deren Inhalte zugreifen zu können. Beispielsweise soll aus einer Menge von Fahrmanövern eine besonders relevante Teilmenge ausgewählt werden, um diese zu visualisieren. Aus diesem Grund wurde die *SPARQL Protocol and RDF Query Language* (SPARQL, [PS08]) als Anfragesprache an RDF-Graphen entwickelt (vgl. [AGP10]), welche im Folgenden in Anlehnung an [HKR08, Kap. 7, S. 201 ff.] eingeführt wird. Die finale Fassung von SPARQL wurde im Januar 2008 vom W3C verabschiedet. Der Einsatz von SPARQL für Wissensbeschreibungen auf

Basis von RDF ist vergleichbar mit dem für SQL auf relationalen Datenbanken, daher ist es wahrscheinlich, dass die Bedeutung von SPARQL in Zukunft noch stark zunehmen wird (vgl.a. [Rit10]).

Damit adressiert SPARQL das Formulieren komplexer Fragestellungen, das Formatieren und Nachbearbeiten der Resultate und Filtern der Ergebnisse. Anfragen orientieren sich an Graph-Mustern (vgl. Abschnitt 2.3.1.2). Zur Ausführung von SPARQL-Anfragen werden diese (ähnlich wie SQL) erst auf eine Algebra abgebildet, um effizient und wohldefiniert berechnet werden zu können (vgl. [AGP10, Kap. 13.3, S. 289 ff.]). Weiterführende Leistungsmerkmale (z.B. Abfragen mehrerer Graphen mit `FROM` [PS08, Kap. 8.2], Schnittstelle für ein Protokoll auf Basis von WSDL [CFT08], Ergebnisdarstellung in XML [BB08]) werden in der Literatur behandelt.

2.3.5.1 SPARQL-Anfragen

Eine Anfrage besteht aus einem Muster für einen Teilgraphen, für den das Enthaltensein überprüft wird. SPARQL orientiert sich an der Turtle-Syntax (vgl. Abschnitt 2.3.1.3), erlaubt jedoch die Verwendung von Variablen. Eine Variable wird durch ein vorangestelltes `?` oder `$` gekennzeichnet.

```
1 PREFIX ex: <http://example.org/>
2 SELECT ?fzg ?manoever
3 WHERE
4 {
5     ?fzg     ex:faehrtManoever     ?x .
6     ?x       ex:Manoever           ?manoever .
7 }
```

Listing 2-4: SPARQL-Abfrage zur Ermittlung von Fahrzeugen und durchgeführten Manövern

In Listing 2-4 ist eine SPARQL-Anfrage dargestellt, welche die Fahrzeuge und die von ihnen ausgeführten Manöver basierend auf dem Graphen in Abb. 2-11 anfragt. Im WHERE-Teil der Anfrage wird das zu suchende Graph-Muster beschrieben, wobei `?fzg`, `?manoever` und `?x` als Variablen dienen. Für alle erfolgreichen Treffer des Graph-Musters gibt der SELECT-Teil der Anfrage an, welche der Variablen als Tupel in das Ergebnis übernommen werden sollen.

Bei der Abfrage von Blank Nodes ist zu beachten, dass im Ergebnis dargestellte Blank Nodes Identifikationsnummern zugeordnet bekommen, die nicht zwangsweise mit denen des zu durchsuchenden Graphen übereinstimmen. Wird ein Literal in einer Anfrage als Objekt eines Tripels explizit angegeben, muss beachtet werden, dass das Graph-Muster nur dann erfolgreich gefunden wird, wenn das Literal exakt übereinstimmt. Jedoch unterstützt SPARQL für einige bekannte Datentypen eine Interpretation des Literals, sodass eine als `xsd:integer` angegebene Zahl 1 als identisch zu einer 01 erkannt werden kann. Als syntaktische Abkürzung für Zahlen ist es außerdem möglich, auf die sonst für ein Literal notwendigen Anführungszeichen mit Datentypangabe zu verzichten.

2.3.5.2 Weitere Sprachmittel von SPARQL

Ein wichtiges Konstrukt für SPARQL-Anfragen ist die Verwendung von Gruppierungen mittels geschweifter Klammern für Teile des gesuchten Graph-Musters. In Kombination mit dem Schlüsselwort `OPTIONAL` werden so Teilgraphen gekennzeichnet, die nicht notwendigerweise gefunden werden müssen. Zwei gruppierte Graph-Elemente lassen sich außerdem über das Schlüsselwort `UNION` verbinden, was einem logischem *Oder* entspricht.

Für einen gruppierten Teilgraphen können weiterhin mit dem Schlüsselwort `FILTER` zusätzliche Filterbedingungen angegeben werden. Bei diesen Bedingungen ist es beispielsweise möglich, Zahlenwerte über Vergleichsoperatoren miteinander in Relation zu setzen. Außerdem können mittels vordefinierter Funktionen bestimmte Eigenschaften der einzelnen Graph-Elemente überprüft werden (vgl. [PS08, Kap. 11]). Auch können die vier Grundrechenarten (Addition, Subtraktion, Multiplikation, Division) so bei der Verknüpfung von Zahlen mit einbezogen werden. Das Beispiel in Listing 2-5 zeigt im Vergleich zu 2-4 die Verwendung optionaler Teilgraphen und Filter.

```
1 PREFIX ex: <http://example.org/>
2 SELECT ?fzg ?manoever ?name
3 WHERE
4 {
5     ?fzg     ex:faehrtManoever     ?x .
6     ?x       ex:Manoever           ?manoever .
7     OPTIONAL { ?manoever ex:hatNamen ?name }
8     FILTER ( langMatches( lang( ?name ), "de" ) )
9 }
```

Listing 2-5: Beispiel für Filter und optionale Graph-Muster in SPARQL

2.3.5.3 Formatierung der Ausgabe

Um die berechneten Ergebnisse den eigenen Bedürfnissen entsprechend aufzuarbeiten und auszugeben, existieren in SPARQL *Modifikatoren*. Hierunter fällt beispielsweise das Sortieren der Ergebnisse unter Verwendung des Schlüsselwortes `ORDER BY`. Die Sortierreihenfolge lässt sich mit `ASC` oder `DESC` als aufsteigend oder absteigend festlegen.

Weiterhin erlauben es die Schlüsselwörter `LIMIT` und `OFFSET`, nur eine Teilmenge des Ergebnisses darzustellen und über die Anzahl der enthaltenen Ergebnistupel zu beschränken. Als weiterer Modifikator ist `DISTINCT` zu nennen, welcher direkt nach dem Schlüsselwort `SELECT` aufgeführt wird. Durch die Verwendung von `DISTINCT` werden in der Ergebnismenge mehrfach auftretende Tupel auf einen einzigen Tupel reduziert.

Zusätzlich zu dem bisher ausschließlich behandelten Ausgabeformat in Form von Tabellen, welches über `SELECT` gewählt wird, lassen sich alternative Ausgabeformate über die Schlüsselwörter `ASK`, `CONSTRUCT` und `DESCRIBE` wählen. Die Verwendung von `ASK` liefert als Ergebnis ausschließlich einen Wahrheitswert (*wahr* oder *falsch*), ob das in der Anfrage beschriebene Baumfragment in dem zu durchsuchenden Graphen vorhanden ist. Das Ergebnis einer `CONSTRUCT`-

oder `DESCRIBE`-Anfrage ist jeweils wieder ein Graph. Eine `CONSTRUCT`-Anfrage erlaubt es, Variablen in einem beliebigen, zu spezifizierenden Graphen darzustellen. Die `DESCRIBE`-Anfrage dagegen liefert üblicherweise die Tripel, welche ein Subjekt besitzen, das einer angefragten Variable entspricht. Alternativ erlaubt eine `DESCRIBE`-Anfrage die direkte Angabe einer konstanten Ressource, um deren beschreibende Aussagen abzurufen.

2.3.5.4 SPARQL und OWL

SPARQL wurde als Anfragesprache für RDF-Graphen entwickelt. Da sich OWL-Dokumente immer als gültiges RDF-Dokument und somit als RDF-Graph darstellen lassen, liegt eine Anwendung von SPARQL auf einem als Graph dargestellten OWL-Dokument nahe.

Die Auswertung von logischen Schlussfolgerungen ist jedoch grundsätzlich *nicht* Bestandteil von SPARQL. SPARQL arbeitet lediglich auf einem zur Verfügung gestellten Graphen ohne ihn zu manipulieren oder zu interpretieren. Allerdings ist es möglich, einen *Reasoner* dazwischenzuschalten (vgl. Abschnitt 2.3.6.2), der einen gegebenen Graphen um geschlussfolgerte Aussagetripel erweitert, und auf dem so erweiterten virtuellen Graphen mittels SPARQL zu arbeiten.

Auf einem so erweiterten Graphen lassen sich jedoch beispielsweise *direkte* Unterklassen von abgeleiteten Unterklassen nicht mehr unterscheiden. Um solche Fragen beantworten zu können, beschäftigen sich diverse Forschungsarbeiten mit Abfragesprachen explizit für OWL, z.B. [FHH04, SP07, KS08b]. Eine ähnliche Akzeptanz wie SPARQL besitzt jedoch zurzeit noch keine dieser Sprachen.

2.3.6 Werkzeuge

Die vorgestellten semantischen Technologien sind nur dann sinnvoll einzusetzen, wenn sie als Bibliothek oder Framework in eigene Software integriert werden können. Bei Semantic Web Werkzeugen ist der Trend festzustellen, dass sich ein Großteil dieser Softwareentwicklungen nur noch auf moderne Programmiersprachen konzentriert und besonders viele Aktivitäten mit der Programmiersprache Java umgesetzt werden.

Mit dem *Jena Semantic Web Framework* [CDD+04, Sou08] und der *OWL API* [HBN07, HB09] stehen zwei weit verbreitete Open-Source-Bibliotheken mit APIs zur Erstellung, Manipulation und Verarbeitung von OWL-Ontologien zur Verfügung. Jena erlaubt außerdem noch die Verwendung von reinem RDF und RDFS, während sich OWL API ganz auf OWL fokussiert.

2.3.6.1 Ontologie-Editoren

Eine besonders öffentlichkeitswirksame Rolle besitzen Ontologie-Editoren für OWL, da sie oft den ersten bewussten Kontakt für Endanwender mit Ontologien darstellen. Sie ermöglichen

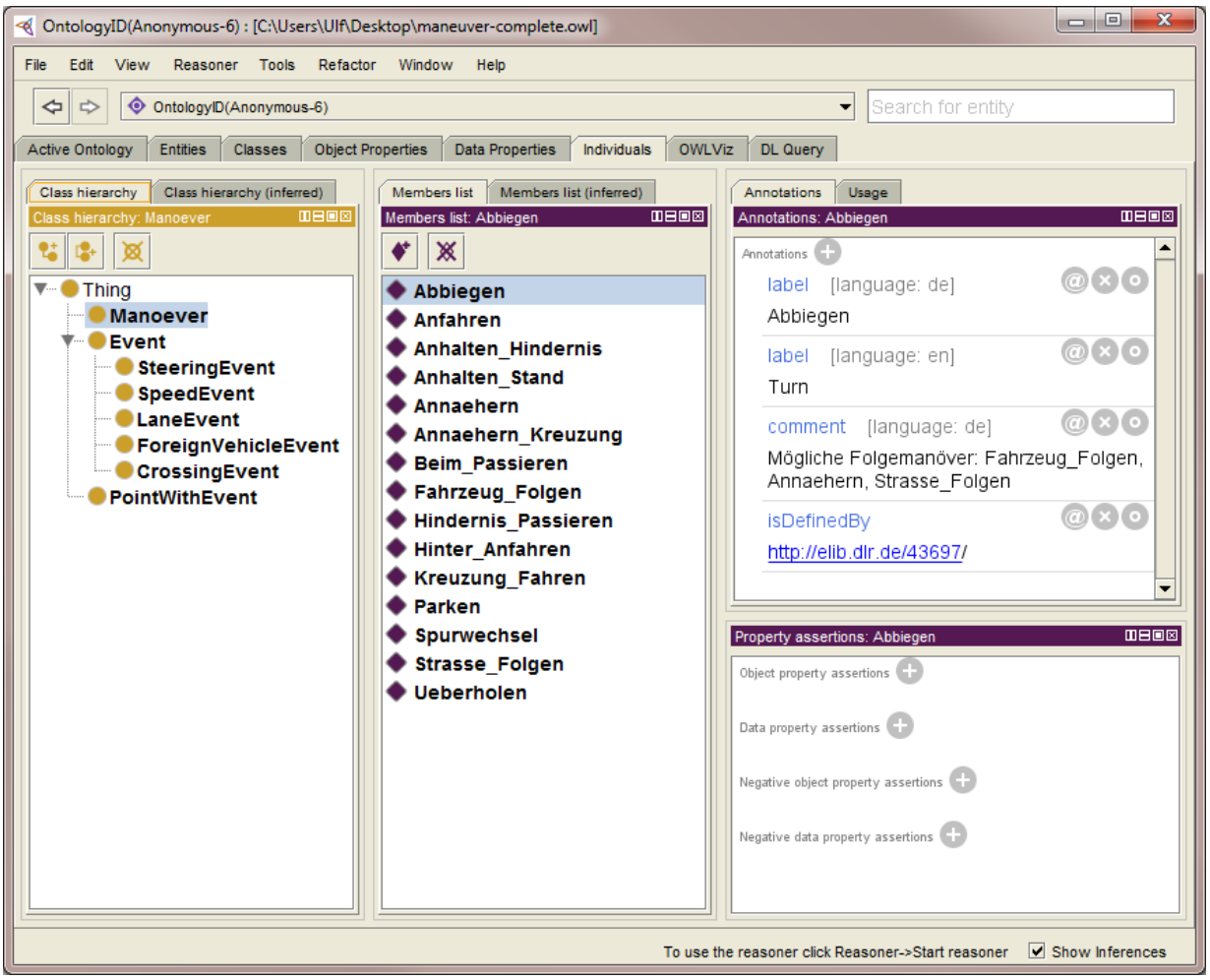


Abbildung 2-14: Protégé beim Bearbeiten einer OWL-Klassenhierarchie[3] (nach [Sta09], Lizenz: [Moz99])

das Erstellen und Bearbeiten von Ontologien, ohne dass sich die Anwender mit dem technischen Hintergrund beschäftigen müssen. Aus akademischer Sicht bieten sie jedoch i.d.R. keine wesentlichen Eigenschaften, die über die zu Beginn dieses Kapitels beschriebenen Fähigkeiten hinausgehen.

Es folgt eine kurze Vorstellung ausgewählter Editoren. Ein besonders populärer Ontologie-Editor ist *Protégé* [GMF+03, Sta09], welcher als freie Software unter der *Mozilla Public License* (MPL, [Moz99]) von den Universitäten Stanford und Manchester entwickelt wird. Als mit am weitesten entwickelte Variante bietet er sämtliche der zuvor aufgeführten Features. Abbildung 2-14 zeigt Protégé beim Bearbeiten einer Klassenhierarchie für Fahrmanöver. Das linke Teilfenster zeigt die Klassen, während das mittlere die Individuen zeigt und rechts die Eigenschaften der Individuen zu sehen sind. *SWOOP* [KPS+06] verfolgt ähnliche Ansätze wie Protégé, wird jedoch nicht mehr aktiv weiterentwickelt. Im Vergleich zu den bereits vorgestellten Editoren ist das *NeOn Toolkit* [HLS+08, Mot10] zum Bearbeiten von Ontologien noch vergleichsweise jung und ist im Rahmen des 6. EU-Forschungsrahmenprogrammes entstanden. Im Gegensatz zu den bisher betrachteten Editoren ist *OwlSight* [PFLT11, Cla10] eine reine Webapplikation. So ermöglicht sie nur das Betrachten, aber nicht das Verändern von OWL-Ontologien.

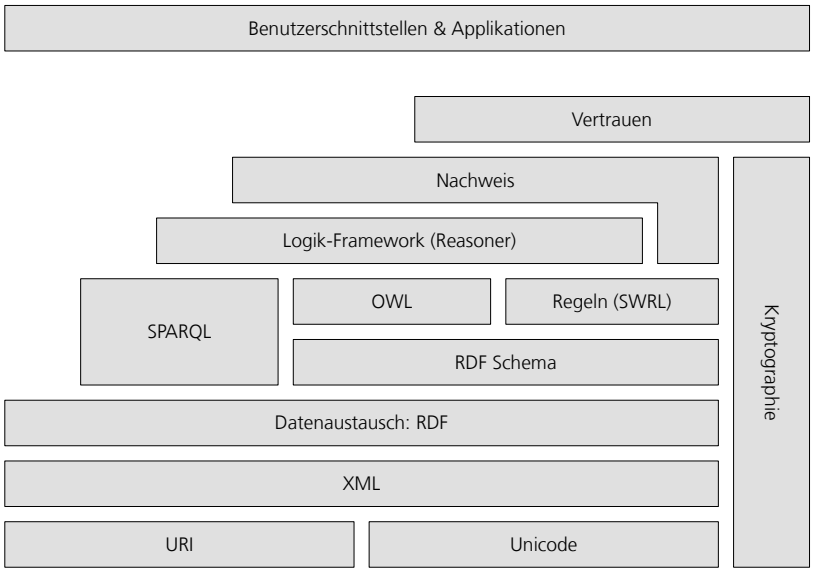


Abbildung 2-15: Semantic Web Stack[2] (frei nach [HPPSH05])

2.3.6.2 Inferenzmaschinen

Inferenzmaschinen (*Reasoner*) erlauben das automatisierte Berechnen von Schlussfolgerungen anhand der in Ontologien modellierten Informationen. Sie werden über Schnittstellen in andere auf RDF und Ontologien basierende Produkte integriert. Reasoner treffen Aussagen über folgende Eigenschaften (vgl. [DLNS94, Kap. 2.3]):

Concept Satisfiability Es wird betrachtet, ob Instanzen einer Klasse existieren können.
Subsumption Es wird geprüft, ob eine Klasse eine andere spezialisiert oder verallgemeinert.
Knowledge Base Satisfiability Eine Ontologie wird auf Widersprüche geprüft.
Instance Checking Für eine Instanz wird geprüft, ob sie zu einer bestimmten Klasse gehört.

Damit erlauben Reasoner neben dem Berechnen von Schlussfolgerungen auch die Überprüfung einer Ontologie auf Konsistenz. Da verschiedene Reasoner unterschiedliche Algorithmen implementieren, weisen sie ein unterschiedliches Geschwindigkeitsverhalten bei der Verarbeitung auf (vgl.a. [GHT06, BHJV08, FU10]). Einige Reasoner verarbeiten insbesondere Inhalte einer ABox besonders schnell, während andere bei der TBox schneller sind (vgl. [BTX+09]). Das Zusammenspiel der einzeln eingeführten semantischen Technologien ist im sogenannten *Semantic Web Stack* (Abb. 2-15) dargestellt.

Die Auswertung logischer Schlussfolgerungen ist eng mit dem Prinzip logischer Programmiersprachen [SHCO95] verwandt. Bekannte Vertreter logischer Programmiersprachen sind das von der *International Organization for Standardization* (ISO) und der *International Electrotechnical Commission* (IEC) standardisierte *Prolog* (ISO/IEC 13211-1, [CR92]) oder *Mercury* [SHC95]. Deduktive Datenbanken basieren ebenfalls auf den gleichen Grundlagen (vgl.a. [GM78, GMN84]).

Mit z.B. *Pellet*, *HermiT*, *FaCT++* oder *KAON2* stehen leistungsfähige Reasoner als freie Software zur Verfügung (vgl.a. [GHT06, BHJV08]). Kommerzielle Beispiele sind *RacerPro* [HM03] oder *Cerebra* [Net04, MK11]. Gerade in diesem Bereich ist die Forschung zurzeit sehr aktiv und neue Algorithmen werden entwickelt. Das *approximative Reasoning* geht von der Annahme aus, dass in dem betrachteten Anwendungsszenario Geschwindigkeit so wichtig ist, dass (temporär) nicht korrekte Schlussfolgerungen erlaubt sind (vgl. [TRKH08]). Diese können nachträglich in einem nachgelagerten Schritt entfernt werden (vgl. [BTX+09]). Mit *Pronto* liegt ein Verfahren vor, mit dem es möglich ist, probabilistische Wissensrepräsentationen in OWL zu verarbeiten (vgl. [Kli08]). Ein Broker-Framework erlaubt die Stärken verschiedener Reasoner zu verknüpfen, indem es je nach Anfrage einen geeigneten zur Verarbeitung auswählt (vgl. [BTX+09]).

2.3.6.3 Speicherung in Datenbanken

Die Speicherung von RDF(S) und OWL erfolgt in spezialisierten Speicherstrukturen oder in relationalen Datenbanken und wird *Triple Store* genannt. So erlaubt ebenso das bereits erwähnte Jena das transparente Speichern dieser Inhalte (vgl. [WSKR03, Sou08]). Ein weiteres RDF-Framework mit speziellem Fokus auf Persistenz ist *OpenRDF Sesame* [AS08, BKvH02].

Der Einsatz von Datenbanken bietet sich allerdings geradezu an, wenn existierende Nutzdaten aus einer Datenbank integriert werden sollen. Die unterschiedlichen Alternativen zur nativen Speicherung von RDF in einer relationalen Datenbank werden in [Vel10] diskutiert.

Grundlegende Idee zur Speicherung von RDF ist die Speicherung einzelner Tripel als Datensatz. Subjekt, Prädikat und Objekt eines Tripels werden in separaten Spalten (Attributen) einer Tabelle (Relation) gespeichert. In der Notation von [KE06] ergibt sich folgende Definition für eine Datenbanktabelle R_{triple_store}:

$$R_{triple\_store} \subseteq D_{subject} \times D_{predicate} \times D_{object} \times D_{object\_type}$$

Subjekte und Prädikate sind eine Teilmenge der URI-Domäne: $D_{subject} \subseteq D_{URI}$, $D_{predicate} \subseteq D_{URI}$. Da Objekte sowohl Blank Nodes, Literale oder URIs sein können, werden sie als $D_{object} \subseteq D_{string}$ gewählt. Für D_{object_type} wird ein Aufzählungsdatentyp gewählt, der den Typ des Objektes angibt. Blank Nodes werden in diesem Schema mittels ihrer Identifikationsnummer dargestellt. Konkrete Implementierungen weichen in der Regel leicht von diesem Schema ab (vgl.a. [WSKR03, BG03]). Diese soeben beschriebene Variante wird beispielsweise von [Sou08] oder [MAD+05] umgesetzt und heißt *Vertical Table* (vgl. [Vel10, Kap. 4.5.2, S. 53 f.]). Ihr Vorteil ist, dass sie unabhängig vom eingesetzten Vokabular ist, keine Annahmen über zukünftige Abfragestrukturen trifft und sich durch die einfache Struktur auf Datenbankebene leicht modifizieren lässt.

Um für bestimmte Szenarien eine verbesserte Suchperformance zu erzielen, existieren mehrere Modifikationen (vgl. [Vel10, Kap. 4.5, S. 50 ff.]). [MAYU05] schlägt die Variante *Graph-based*

Storage vor, bei der statt einfacher Tripel gleich ganze Graph-Pfade gespeichert werden, nach denen mit hoher Wahrscheinlichkeit gesucht werden wird. Der *Graph Schema-Vertical Data* genannte Ansatz basiert auf RDFS und sieht vor, dass die konkrete Struktur aus RDFS in Tabellen überführt und somit von der Instanz-Ebene getrennt wird (vgl. [ACKP01]). Ein ähnlicher Ansatz wird in [VNGP10] für OWL-Ontologien vorgeschlagen. Dieser Ansatz ist vergleichbar mit einem Datenbankschema, wie es ähnlich von *Object-Relational Mapping* (ORM, [BS98]) Werkzeugen realisiert wird. Die in diesem Absatz aufgeführt Varianten zur Speicherung von RDF(S) bieten zwar unter ganz bestimmten Bedingungen eine verbesserte Performance, jedoch wird durch diese Ansätze das Datenbankschema deutlich komplexer.

Eine vergleichsweise neue, direkt auf *Vertical Data* aufbauende Variante heißt *Smart Indexing* (vgl. [WKB08, NW08]). Bei diesem Verfahren werden die Aussagen wie zuvor als Tripel gespeichert, jedoch wird für jeden der $3! = 6$ möglichen Zugriffspfade der Tripel ein Index angelegt. So wird auf Kosten eines erhöhten Speicherbedarfes und der Zeit zur Erstellung des Indexes eine verbesserte Abfrageperformance erzielt. Diese Variante bietet den Vorteil, dass der Index transparent vom DBMS verwaltet werden kann und das Datenbankschema einfach bleibt, sodass sich die RDF-Aussagen auf Datenbankebene weiterhin leicht ändern lassen. Außerdem sind die Datenbankstrukturen unabhängig von der Struktur eines RDFS-Dokumentes oder einer Ontologie. Bisher existieren wenigstens die zwei Implementierungen *RDF-3X* und *Hexastore* für diesen Ansatz.

Ein anderes Ausgangsproblem liegt dem *Mapping* von relationalen Daten in RDF(S) bzw. OWL zugrunde. Hierbei werden schon vorhandene Datenbestände aus einer Datenbank in RDF konvertiert (vgl. [BL98, BG03]). Um dieses Mapping zu realisieren, existieren Frameworks (z.B. *R2O* [BOCGP04] oder *D2R Map* [Biz03]), die auf Regeln oder Vorlagen basieren, anhand derer die Konvertierung beschrieben wird. Für Protégé existiert ebenfalls ein Plugin, welches den Import von Daten aus einer relationalen Datenbank in eine Ontologie erlaubt (vgl. [NOT07]). Mit *OpenLink Virtuoso* wird eine Software angeboten, welche sowohl die Aufgaben eines Triple-Stores als auch das Mapping von relationalen Daten in RDF ermöglicht und somit als Datenintegrationsplattform beworben wird (vgl. [EM10]). Das Mapping relationaler Daten ist jedoch für Messdaten ungeeignet (vgl. Abschnitt 3.5.1) und somit für diese Arbeit nicht relevant.

2.3.7 Anwendungsfälle

Ein durch die Verwendung von Ontologien angestrebtes Ziel ist die strukturierte Wissensdarstellung und deren Wiederverwendung, damit die betrachteten Anwendungsfälle zueinander in Beziehung gesetzt werden können. Abbildung 2-16 zeigt eine Übersicht über die Vernetzung bereits existierender, wichtiger Ontologien. Im Zentrum mit den meisten Verknüpfungen befindet sich die *DBpedia* [ABI11], welche das Ziel hat, in der *Wikipedia* gespeicherte Fakten in Form einer Ontologie zur Verfügung zu stellen (vgl. [ABK+07]). Anhand der Größe der Kreise ist die Größe der Ontologien zu erkennen. Die kleinsten Ontologien enthalten zwischen 1.000 und

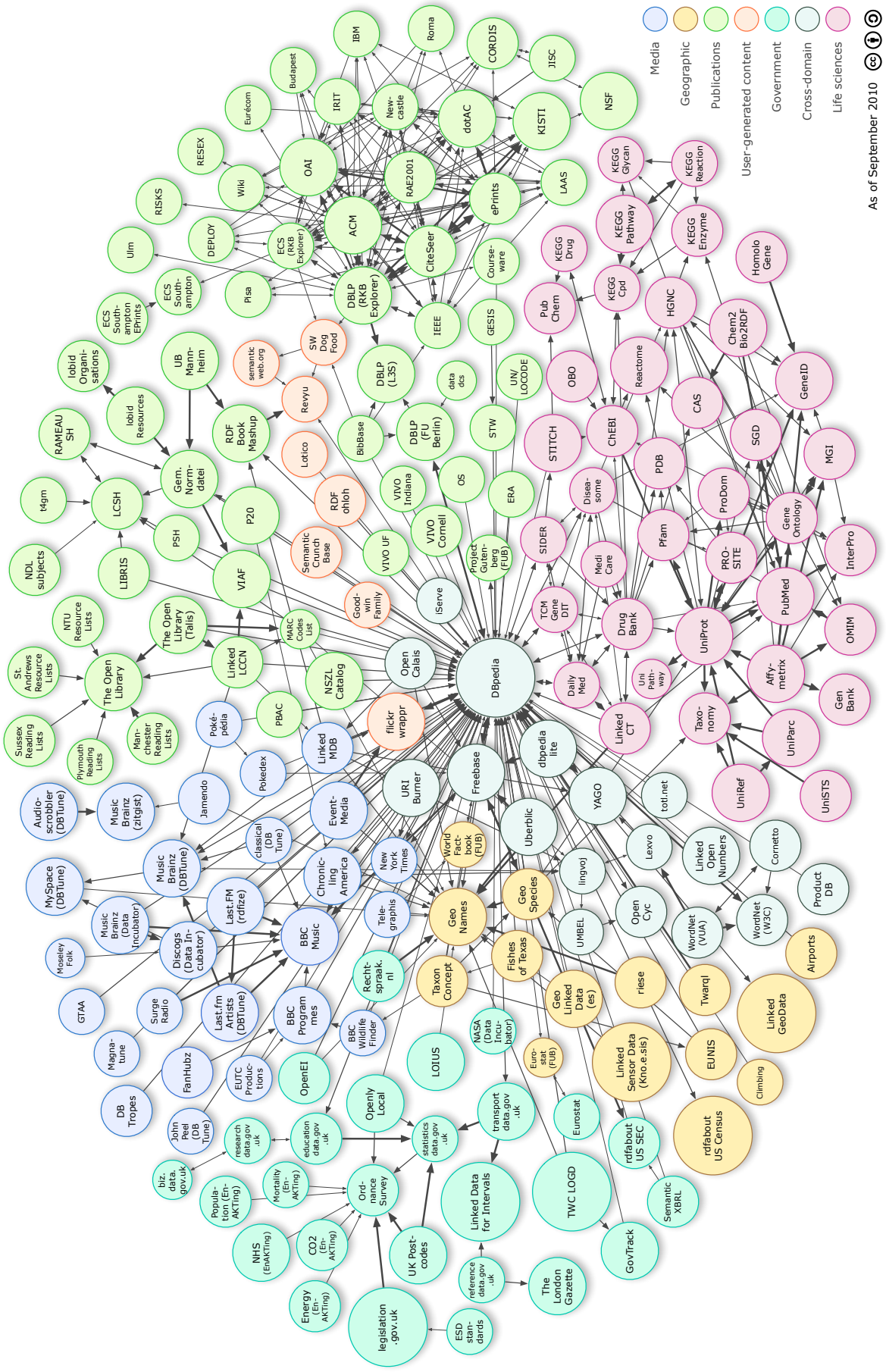


Abbildung 2-16: Linking Open Data Cloud Diagram[1], Richard Cyganiak und Anja Jentzsch. http://lod-cloud.net

10.000 Tripeln. Die größten Kreise repräsentieren Ontologien mit mehr als 1 Milliarde Tripeln. Die Dicke der verknüpfenden Pfeile repräsentiert die Anzahl der Verknüpfungen zwischen den Ontologien, wobei die dicksten Pfeile mehr als 100.000 Verknüpfungen darstellen.

In [SS09, Teil 3 u. 6, S. 359 ff. u. 711 ff.] wird eine Übersicht zum Einsatz von Ontologien und semantischen Technologien in diversen verschiedenen Szenarien vorgestellt. Eines der ersten und bekanntesten Vokabulare überhaupt stellt der *Dublin Core* [Dub09] dar. Er wurde entworfen, um als Grundlage für das Semantic Web WWW-Seiten und deren Inhalte zu beschreiben. Auch im Bereich des *Social Web* kommen semantische Technologien zur Beschreibung von Beziehungen zwischen Individuen zum Einsatz (vgl. [BP09]). Beim Einsatz von *Web Services* zum Zugriff auf fremde Funktionen wird mit semantischen Technologien deren Funktionalität beschrieben, sodass geeignete Webservices automatisiert gefunden und eingebunden werden können (vgl. [PLL06]).

Im Bereich der Datenbanken helfen Ontologien, deren Schema zu beschreiben, um so das Mapping verschiedener Datenbanken aufeinander zu unterstützen oder Datenbanken mit unterschiedlichem Schema und äquivalenter Bedeutung einheitlich zu behandeln (vgl.a. [KS96, ANP10, Köh03]). Die Struktur von auf Datenbanken aufbauenden Data-Warehouses kann ebenfalls mit Hilfe von Ontologien abgebildet werden (vgl.a. [HW08, Net04]). Im Informationsmanagement der Systembiologie [Eck11] und in der Bioinformatik [EEM+07] dienen Ontologien nicht nur zur Schemabeschreibung und Integration verschiedener Datenbanken, sondern auch um die Evolution des Datenbankschemas nachzuverfolgen (vgl. [KES+07]). Für diesen Zweck wird zu einem Datenbankschema zunächst eine beschreibende Ontologie generiert. Ändert anschließend eine Operation (`CREATE`, `ALTER`, `DELETE`) das Schema, wird ebenfalls die entsprechende Ontologie synchronisiert, sodass das alte und das neue Schema und deren Beziehung zueinander dokumentiert sind. Es werden jedoch keine Aussagen über die in der Datenbank enthaltenen Datensätze getroffen, wie es diese Arbeit anstrebt.

Inhaltlich beschäftigen sich Ontologien mit sehr unterschiedlichen Domänen. Auch zu Bereichen, die sich direkt oder indirekt mit dem Thema Verkehr beschäftigen, existieren einige wenige Arbeiten. So wird in [RKT05] die Modellierung von Sensoren betrachtet und [SW08] stellt Aspekte für eine Ontologie zur Beurteilung der Leistung von unbemannten Robotern vor. Weiterhin existieren Arbeiten zur Modellierung des Fahrerverhaltens [BCH+08] oder für die Beschreibung von Kreuzungen [HYD07, HTL07]. Über die ontologiebasierte Systementwicklung für satellitengestützte Positionsbestimmungssysteme wird in [BHMS09] berichtet.

Steht für einen gewünschten Anwendungsfall keine hinreichende bereits existierende Ontologie zur Verfügung, kann sie mit den in Abschnitt 2.3.6.1 vorgestellten Werkzeugen selber modelliert werden. Diese stellt dann für die betrachtete Domäne notwendige wiederkehrende Eigenschaften und Begriffe bereit. In einer Ontologie liegt immer eine Chance für einen Anwendungsfall, alle relevanten Begriffe zu systematisieren und unter allen beteiligten Projekt-Stakeholdern ein einheitliches Vokabular zu etablieren. Auf diese Weise trägt ein entsprechen-

des Begriffssystem zu einer verbesserten Zusammenarbeit bei und vermeidet Missverständnisse aufgrund sprachlicher Mehrdeutigkeiten.

2.3.7.1 Zeit in OWL-Ontologien

Die Sprachmittel semantischer Technologien sind ursprünglich zur Modellierung von statischem Faktenwissen ausgelegt. Daher müssen bei der Beschreibung dynamischer Vorgänge, bei denen sich Beziehungen im Laufe der Zeit ändern, die temporalen Eigenschaften explizit modelliert werden (vgl. [WFM06]). So ist die Betrachtung dynamischer Eigenschaften in Ontologien derzeit ein viel diskutiertes Forschungsthema, jedoch noch nicht abschließend gelöst (vgl.a. [KKD08, KVH08, THOD09]).

Anwendungsfallunabhängig betrachtet können zeitliche Angaben zu einer Ressource ggf. mit in der URI der Ressource gespeichert werden (vgl. [PPP05]). Jedoch bietet sich alternativ eine explizite Modellierung in Ontologien durch Einsatz entsprechender Entwurfsmuster [GP09] an. Auch die Einbindung und Berücksichtigung spezieller Ontologien, mit deren Hilfe zeitliche Aspekte modelliert werden können, ist eine alternative Möglichkeit. Für diesen Einsatzzweck wurde die *tOWL*-Ontologie entworfen (vgl. [FMK10]). Sie vereinigt die Entwürfe von *Temporal RDF* [CG07], *OWL-Time* [HP06] und *4D-Fluents* [WFM06].

Beim Einsatz von tOWL wird zwischen drei Repräsentationsebenen unterschieden. Die grundlegendste ist die der eigentlich zu betrachtenden Domäne. Darauf setzt die Ebene der zeitlichen Repräsentation der zu betrachtenden Elemente auf. Und wiederum hierauf basiert die Repräsentation von Zustandsänderungen der Elemente (vgl. [WFM06]). Für die temporale Repräsentation werden konkrete Zeitangaben in Form von Zeitpunkten und Intervallen eingeführt. Die Intervalle werden über die 13 Allen-Relationen [All83] (Abschnitt D.13) zueinander in Beziehung gesetzt. Jedoch erlaubt tOWL mit einem Standard-OWL-Reasoner lediglich die Überprüfung der Konsistenz der betrachteten Ontologie.

Die Modellierung temporaler Phänomene besitzt neben den logischen Herausforderungen auch vielschichtige weitere Dimensionen (z.B. Kausalität, Zeitempfinden, Philosophie, . . .). Bei der alleinigen Betrachtung auftretender Ereignisse, kann es daher sinnvoll sein, die Zeit gar nicht explizit zu modellieren und stattdessen nur die Beziehung benachbarter Ereignisse zu betrachten. Dieser Ansatz wird im Rahmen der vorliegenden Arbeit verfolgt (vgl. Abschnitt 3.5).

2.3.8 Vergleich und Bewertung semantischer Technologien

Die Möglichkeit, aus einem Modell implizites Wissen abzuleiten, steigt mit dem Umfang der axiomatischen Aussagen und Konventionen für die Semantik der eingesetzten Modellierungssprache (vgl. Abschnitt 2.3.1.4, 2.3.2.3, 2.3.3.4). Die eigentliche Leistung von RDF liegt in der Bereitstellung des leistungsfähigen und gemeinsamen Datenmodells. Erst die Einführung von

Klassen- und Eigenschaftenhierarchien mit der Trennung von Schema- und Instanzwissen in RDFS erlaubt eine Ontologie-Modellierung mit umfangreichen Reasoning. Beispielsweise können über die Hierarchien Spezialisierungsaussagen abgeleitet werden und die Domain- und Range-Angaben erlauben eine Konsistenzprüfung der verwendeten Eigenschaften.

Unter OWL werden Klassen als Mengen von Individuen betrachtet, für die logische Mengenbeziehungen (disjunkt, komplementär etc.) eingeführt werden, sodass eine Beschreibung ähnlich der Prädikatenlogik möglich wird, welche formal nachvollzogen werden kann. Die eingeführten Attribute für Eigenschaften (transitiv, symmetrisch etc.) ergeben weitere neue Möglichkeiten für das Reasoning. Schließlich erweitert SWRL die OWL um die freie Definition von Regeln, welche durch einen Reasoner umgesetzt werden. Bei den Regeln besitzt der Anwender große Freiheit, lediglich die Wahrung der Konsistenz für die betrachtete Ontologie darf bei der Anwendung der Regeln nicht verletzt werden.

Obwohl Zahlen und Zeichenketten als Literal dargestellt werden können, können diese nicht mittels logischer Ausdrücke ausgewertet werden. Diese Möglichkeit besteht erst bei der Formulierung von Abfragen mit SPARQL, wobei die Anfragen aber nicht Bestandteil der Ontologie sind. Erst SWRL erlaubt diese Modellierung mittels integrierter Funktionen (Built-Ins).

Eine praktische Einschränkung von OWL ist, dass über Reasoning für viele Konstrukte lediglich die Klassenzugehörigkeit abgeleitet werden kann. Das Ableiten von Relationen zwischen Individuen ist mittels der Eigenschaften (invers, symmetrisch etc.) und Ketten für Rollen vergleichsweise eingeschränkt. Erst SWRL überwindet diese Einschränkung mit frei definierbaren Regeln.

Die in diesem Abschnitt vorgestellten semantischen Technologien besitzen klar abgegrenzte Leistungsmerkmale und bauen aufeinander auf, was auch durch den Semantic Web Stack (Abb. 2-15) deutlich wird. Daher stellen sie ihr volles Leistungsvermögen erst zur Verfügung, wenn sie gemeinsam eingesetzt werden. In dieser Kombination können sie für ein umfassendes Wissensmanagement eingesetzt werden, so wie dies zur umfangreichen Beschreibung von Messdaten wünschenswert ist. Semantische Technologien stellen somit eine sehr geeignete Grundlage dar, um die zuvor behandelten Methoden zum Versuchsdatenmanagement zu ergänzen.

2.4 Zusammenfassung

In dem aktuellen Kapitel werden zuerst Arbeiten mit Bezug zur behandelten Fragestellung vorgestellt. Der Fokus liegt zunächst auf datenbankzentrierten Ansätzen zur Verwaltung wissenschaftlicher Versuchsdaten. Im Anschluss werden Systeme behandelt, deren Fokus auf der Durchführung von Verarbeitungsschritten auf wissenschaftlichen Daten liegt. Danach werden Szenarien betrachtet, in denen die Inhalte von Datenprodukten auf Datenelementebene semantisch mit Annotationen beschrieben werden.

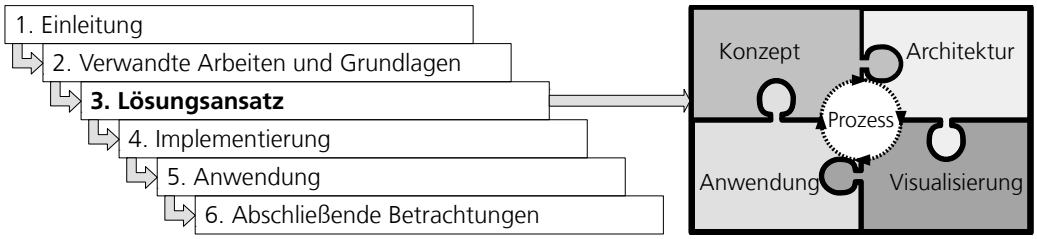
Datenbanken bilden eine solide Grundlage zum Speichern großer Datenmengen. Aber keines der betrachteten Systeme beschreibt Messdaten auf der Ebene feingranularer Messpunkte mit Metadaten. Auch werden zur Beschreibung keine formalen Modelle eingesetzt, sodass automatisierte Schlussfolgerungen auf den Messdaten nicht durchgeführt werden können. Der Einsatz einer offenen Architektur zur Einbindung externer Datenquellen für die Dokumentation und Beschreibung ist ebenfalls nicht zu finden.

Der Abschnitt 2.3 behandelt ausführlich die Grundlagen semantischer Technologien. Es werden RDF, RDFS, OWL, SWRL, SPARQL und darauf beruhende Werkzeuge erläutert. Mit ihnen lassen sich inhaltliche Zusammenhänge auf einer hohen abstrakten Ebene als formales Modell beschreiben.

Semantische Technologien erlauben somit, aus modellierten Eigenschaften automatisiert Schlussfolgerungen zu ziehen, so implizites Wissen abzuleiten und zu durchsuchen. Weiterhin beruhen sie auf Internetstandards und sind darauf ausgelegt verschiedene externe Quellen einzubinden. Aus diesen Gründen bieten semantische Technologien wesentliche Merkmale, die sich ideal zur Beschreibung von Versuchsdaten eignen.

Die in diesem Kapitel betrachteten Arbeiten zeigen somit, dass Datenbanken und semantische Technologien sich gegenseitig ideal ergänzen können, um Versuchsdaten auf feingranularer Ebene formal zu beschreiben. Im folgenden Kapitel werden aufbauend auf diesen Grundlagen die neuartigen Konzepte eingeführt, welche die detaillierte Beschreibung von Versuchsdaten mit semantischen Technologien ermöglichen. Diese Kombination erlaubt, die Vorteile der jeweiligen einzelnen Lösungen zu steigern. Die vorliegende Arbeit leistet durch die vorgestellten Ansätze daher einen wesentlichen Beitrag zur Verbesserung des Managements von Versuchsdaten.

3 Lösungsansatz



Das vorhergehende Kapitel behandelt aktuelle Ansätze zur Verwaltung von Messdaten aus wissenschaftlichen Versuchen. Eine formale Beschreibung auf feingranularer Ebene zur Beschreibung einzelner Messpunkte ist derzeit leider nicht zu finden. Mit einem solchen Ansatz ist es jedoch möglich, komplexe Anfragen für Messpunkte zu stellen, formale Modelle einzubinden, Schlussfolgerungen aus den Messdaten zu ziehen und externe Quellen zur Beschreibung einzubinden. Beispielsweise soll es so möglich werden, Fahrmanöver in aufgezeichneten Fahrexperimenten zu modellieren.

Der Aufbau dieses Kapitels orientiert sich an den verschiedenen Aspekten der Forschungsfrage (Abschnitt 1.4), wie in einer Datenbank gespeicherte numerische Daten mit formalen Modellen in Form von Ontologien beschrieben werden können. Zuerst wird ein neuartiges *Konzept* entwickelt (Abschnitt 3.1), um diese Ziele zu erreichen und die in einer relationalen Datenbank gespeicherten Messdaten mit semantischen Technologien zu verknüpfen. Im Rahmen des Konzeptes werden die angestrebten Verknüpfungsmöglichkeiten zwischen Zeitreihen und semantischen Technologien vorgestellt, um sie in definierten Anwendungsszenarien nutzbringend einsetzen zu können. Als Grundlage für die Umsetzung wird eine *Architektur* entworfen (Abschnitt 3.2), welche den aufgezeigten Bedürfnissen gerecht wird. Für die Umsetzung wird weiterhin der *Prozess* berücksichtigt (Abschnitt 3.3), damit ein reibungsloser Arbeitsfluss gewährleistet werden kann. Als Nächstes werden die Konzepte für eine entsprechende *Visualisierung* und Benutzerinteraktion entwickelt (Abschnitt 3.4). Die *Anwendung* erfolgt anhand von einem formalen Automatenmodell zur Beschreibung von Ereignissen (Abschnitt 3.5), um automatisiert Schlussfolgerungen aus diesen zu ziehen. Abbildung 3-1 verdeutlicht das Zusammenspiel dieser Aspekte.

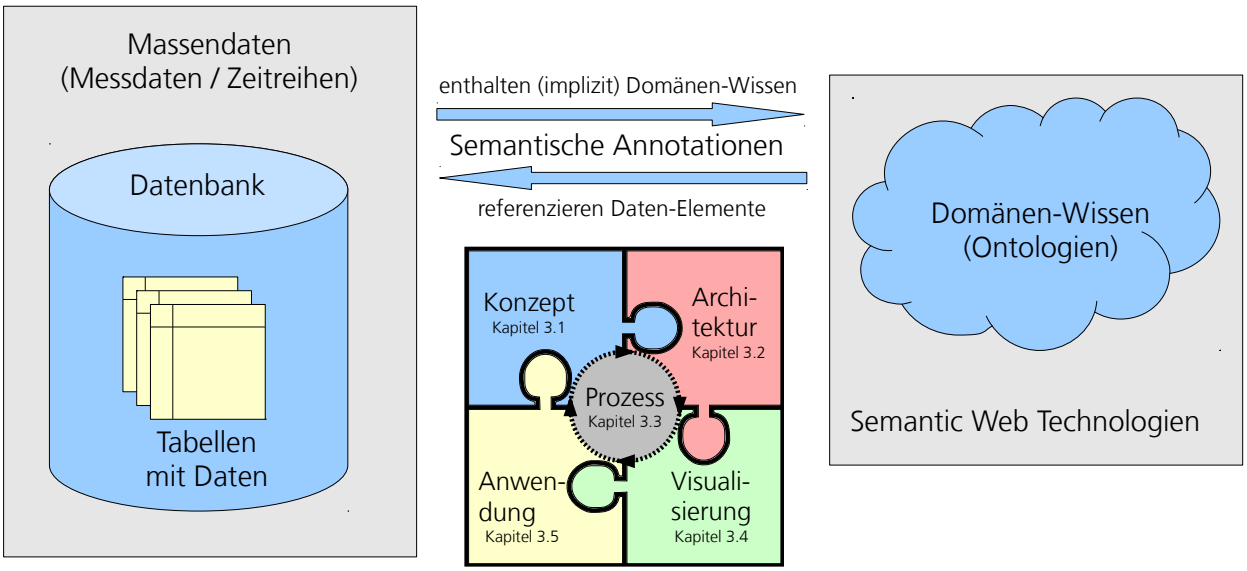


Abbildung 3-1: Einsatz semantischer Annotationen

3.1 Konzept zum Einsatz semantischer Technologien zur Annotation von Datenbank-Elementen

Die in Experimenten erfassten Messdaten der Sensorik sind als Zeitreihen in einer relationalen Datenbank gespeichert. Um eine langfristige Interpretation der Daten und lang andauernder Studien zu ermöglichen, müssen diese mit zusätzlichen Informationen zur Interpretation verknüpft werden. Dieses Szenario muss zum einen Informationen über die ursprünglich erhobenen Zeitreihen umfassen, als auch beliebige im Nachhinein festgestellte Tatsachen über identifizierte Ereignisse in den Daten und Eigenschaften. Die besondere Herausforderung ist die Behandlung *beliebiger* Daten-Annotationen, welche so formal erfasst werden können, dass eine weitgehende maschinelle Verarbeitung möglich ist.

Die Offenheit und Erweiterbarkeit der Daten-Annotationen ist bei diesem Prozess eine besondere Notwendigkeit, da die Anwendung in verschiedenen Betrachtungsebenen häufigen Änderungen unterworfen ist. Zum einen sind Experimentalsysteme, deren Komponenten und deren Konfiguration selbst einer kontinuierlichen Weiterentwicklung unterworfen, sodass sich evolutionär weiterentwickelnde Annotationen auftreten. Zum anderen sind die Experimente i.d.R. Teil neuer, individueller Projekte, welche unbekannte Fragestellungen mit sich bringen und so verschiedene Wissensdomänen berühren. Auch die Bearbeitung in internationalen Teams mit unterschiedlichen Sichtweisen erfordert leicht zu ergänzende Daten-Annotationen mit der Möglichkeit, bereits vorhandenes Wissen zu integrieren.

Abbildung 3-1 zeigt die grundlegende Herangehensweise des in dieser Arbeit behandelten Ansatzes. Es werden zum einen Datenbanktechnologien (links) zur Speicherung von großen Datenmengen mit semantischen Technologien (rechts) zur Beschreibung von abstrakten Inhalten (z.B. Fahrmanöver) miteinander kombiniert, um von deren jeweiligen Vorteilen zu profitieren.

Die Aspekte *Konzept*, *Architektur*, *Visualisierung* und *Anwendung* betrachten den kombinierten Einsatz aus verschiedenen Perspektiven. Die genannten Aspekte sind wesentlicher Bestandteil der Forschungsfrage, welche der Arbeit zugrunde liegt, und spiegeln sich im Aufbau der Arbeit wider. Als verknüpfende Elemente werden *semantische (Datenbank-)Annotationen* (vgl. [NBK11, BL98]) eingesetzt (vgl.a. [KKPS01, HSWW03, BPS+05]). Diese Annotationen werden durch diese Arbeit neu eingeführt und bilden eine entscheidende Grundlage für die folgenden Betrachtungen.

Definition 4 (semantische Annotation) *Eine semantische (Datenbank-)Annotation ist ein Aussagetripel aus Subjekt, Prädikat und Objekt* $(s, p, o) \in S$ *eines RDF-Modells* $M = (R, P, \mathcal{L}, S)$ *(siehe Def. 3), für welches gilt, dass* $s \in R$ *ein Datenbank-Element repräsentiert oder dass* $o \in R$ *ein Datenbank-Element repräsentiert. Der Tripel* (s, p, o) *verknüpft somit ein Datenbank-Element mit einem Wissensmodell bzw. einer Ontologie, um ein Datenbank-Element in einen inhaltlichen Bezug zu setzen.*

3.1.1 Datenbank-Elemente für Annotationen

Als Basis für die Betrachtungen diene ein Satz Messdaten, welche in tabellarischer Form vorgehalten werden. Diese sollen mit zusätzlichen Informationen verknüpft werden können. Zu diesem Zweck werden Verknüpfungen bzw. Annotationen mit verschiedenen Bezügen eingeführt, welche in Abb. 3-2 dargestellt werden.

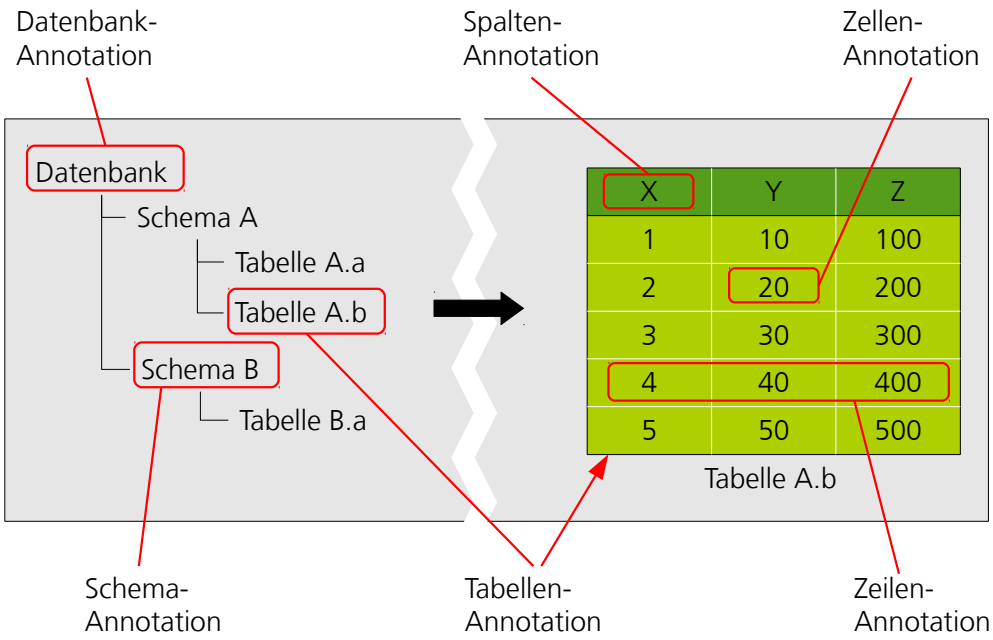


Abbildung 3-2: Tabellen einer relationalen Datenbank und mögliche Annotationen[2] (nach [NBK11])

Datenbank Die gesamte relationale Datenbank umfasst sämtliche Tabellen und kann ebenfalls das Ziel von Annotationen sein. Ein mögliches Annotationsbeispiel sind Zuständigkeiten für administrative Tätigkeiten.

Schemas Sie organisieren Tabellen ähnlich wie Verzeichnisse für Dateien, jedoch lassen Schemas sich nicht schachteln. Sind die Messungen einzelner Kampagnen in Schemas organisiert, lassen sich an den Schemas Annotationen mit Informationen zur durchgeführten Kampagne vermerken.

Tabellen Das Tabellen-Schema von Spalten und ihren Eigenschaften (Datentyp etc.) einer Tabelle wird als gegeben und unveränderlich betrachtet. In einer Tabelle werden zusammenhängende Daten gespeichert, wie z.B. die zu einem einzelnen Experiment gehörenden Messdaten. Mögliche Annotationen für Tabellen können z.B. Informationen zum durchgeführten Experiment oder zu einem Sensor umfassen, welcher die Daten der Tabelle erfasst hat.

Spalten Sie werden auch Attribute genannt. Spalten besitzen einen Namen und einen Datentyp und können in einer Datenbank mit zusätzlichen Bedingungen zu den enthaltenen Daten versehen sein (Constraints). Somit sind die Inhalte einer Spalte immer von einer einheitlichen Struktur und treffen in jeder Zeile eine Aussage über *etwas* vom gleichen Typ. Annotationen bezüglich einer Spalte können beispielsweise die physikalische Einheit eines Messwertes oder eine textuelle Beschreibung des Spalteninhaltes sein.

Zeilen Sie werden auch Datensätze oder Records genannt. Für die vorliegende Arbeit wird die einschränkende Annahme getroffen, dass sich jede Zeile, auf die Bezug genommen werden soll, eindeutig identifizieren lässt. Dies wird durch einen eindeutigen Wert in einem Attribut erreicht (z.B. laufende Nummer oder eindeutiger Zeitstempel). Beispiel-Annotationen für Zeilen beschreiben ein aufgetretenes Ereignis (z.B. Spurwechsel).

Zellen Sie werden auch als Attribut-Instanz bezeichnet. Eine Zelle wird durch die Spalte und die Zeile, in der sie liegt, identifiziert. Die Annotation einer Zelle bezieht sich somit auf einen einzelnen Wert. Sie kann z.B. die Qualität dieses einzelnen Wertes dokumentieren oder vermerken, dass der Wert händisch korrigiert wurde.

Weitere Beispiele für konkrete Annotationen auf Basis der einzelnen Datenbank-Elemente lassen sich aus der Metadaten-Kategorisierung in Abschnitt 2.2.1 ableiten. Im Gegensatz zu den sonst bei Scientific Workflow Systemen (Abschnitt 2.2.3) eingesetzten Metadaten oder der Annotation von Datenbankschemas (Abschnitt 2.3.7) wird somit der Unterschied deutlich, dass sich die Annotationen nicht nur auf komplette Datensätze, sondern auch auf deren feingranulare Inhalte beziehen. Dieser neuartige Betrachtungsgegenstand ist ein wesentlicher Beitrag dieser Arbeit.

Die soeben diskutierte Herangehensweise ist inspiriert von Multimediaannotationen (vgl. Abschnitt 2.2.5) und adaptiert diese auf den Bereich von Messdaten bzw. relationalen Daten (vgl.a. [Kip01, BPS$^+$05, HOCM$^+$05]). Aus diesem Bereich ist der Begriff der Mikro-Annotationen entlehnt. Mikro-Annotationen beziehen sich im Gegensatz zu (normalen) Annotationen auf sehr feingranulare Elemente. Denn im Gegensatz zu anderen Arbeiten zur Verwaltung und Verarbeitung von Messdaten, erlauben die in dieser Arbeit vorgestellten Ansätze und Methoden eine Beschreibung auf der Ebene einzelner Messwerte. Daher werden Annotationen für Zeilen oder Zellen bisweilen als Mikro-Annotationen bezeichnet.

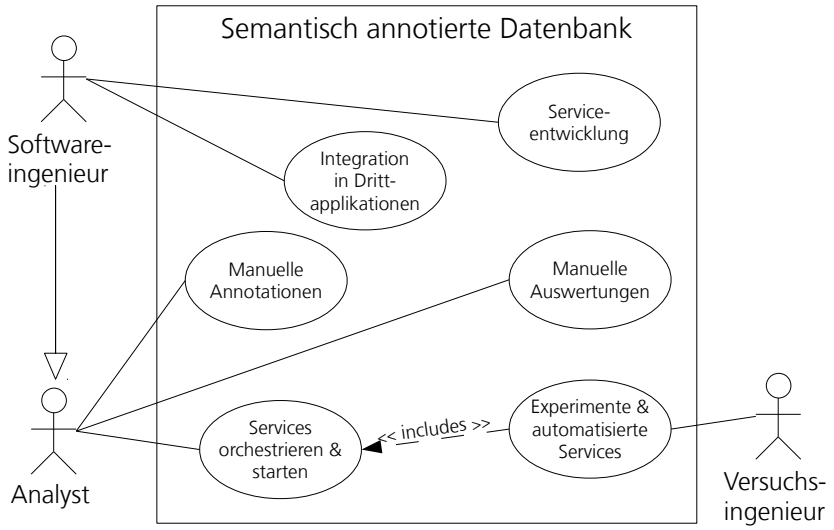


Abbildung 3-3: Anwendungsfalldiagramm zur Verwendung semantischer Annotationen für Datenbank-Elemente

3.1.2 Anwendungsfälle

Für die technische Umsetzung der Datenbank-Element-Annotationen (Def. 4) sollen die getroffenen Aussagen mittels der in Abschnitt 2.3 eingeführten semantischen Technologien umgesetzt werden. Dieses ermöglicht eine formale Umsetzung der getroffenen Aussagen und eine nach dem derzeitigen Stand der Technik möglichst weitgehende Interpretation der Aussagen durch Software. Zur Motivation lassen sich auf der soeben eingeführten Basis verschiedene Anwendungsfälle für den praktischen Einsatz semantischer Annotationen ableiten. Bezüglich der Anwendungsfälle ist zu betonen, dass der Fokus dieser Arbeit auf der Anwendung der vorgestellten Konzepte zur Unterstützung von Analysen liegt. Die Analysen selber können wiederum Teil eines automatisierten Workflows (SWS) sein, was jedoch bei den folgenden Betrachtungen nur eine untergeordnete Rolle spielt.

Abbildung 3-3 zeigt verschiedene Anwendungsfälle zum Einsatz von Annotationen auf Basis semantischer Technologien. Dabei ist unter dem dargestellten System „Semantisch annotierte Datenbank“ eine Datenbank im weiteren Sinne zu verstehen. So müssen die im Folgenden aufgeführten Anwendungsfälle nicht zwingend *in* der Datenbank ablaufen, es ist vielmehr davon auszugehen, dass *auf* die Datenbank zugegriffen wird. Für die Anwendungsfälle werden verschiedene Rollen von Anwendern berücksichtigt, welche im Folgenden beschrieben werden:

Analyst Der Analyst ist der typische Anwender in der Mehrheit der Anwendungsfälle. Er verfügt über ein ausgeprägtes Domänenfachwissen und möchte unter diesem Gesichtspunkt die Daten analysieren. Er betrachtet interaktiv annotierte Daten und fügt ihnen gewonnene Erkenntnisse manuell als Annotationen hinzu. Alternativ wählt er einen Auswertealgorithmus aus einem Angebot von zur Verfügung stehenden aus und veranlasst, dass dieser mit den Daten seiner Wahl entsprechende Berechnungen durchführt. Da diese Auswertealgorithmen ihre Arbeit autonom erledigen, werden sie als *Service* bezeichnet.

Softwareingenieur Der Softwareingenieur ist eine spezialisierte Form des Analysten, der in der Lage ist, Services selbständig zu entwickeln. Neben dem notwendigen Domänenwissen verfügt er somit über das Wissen, Algorithmen zu programmieren und um auf die notwendigen Entwicklerschnittstellen zuzugreifen.

Versuchsingenieur Der Versuchsingenieur überwacht die korrekte Durchführung von Experimenten, welche neue Versuchsdaten zur Auswertung liefern. Die Versuchsdaten werden im Anschluss an das Experiment sofort und automatisiert in der Datenbank gespeichert.

Die einzelnen Anwendungsfälle aus Abb. 3-3 werden im Folgenden erläutert:

Manuelle Auswertungen Der manuelle Zugriff ist die rudimentärste Art, sich mit der Datenbank und ihren Annotationen zu beschäftigen. Für den manuellen Zugriff wird eine Benutzerschnittstelle angeboten, über die dem Anwender die komfortable Exploration der annotierten Daten ermöglicht wird. Idealerweise orientiert sich eine solche Oberfläche an der schematischen Darstellung in Abb. 3-2, sodass die grundlegende Idee und die praktische Umsetzung der Annotationen sich dem Anwender schnell erschließen.

Der Anwender kann mittels der Benutzerschnittstelle direkt die Annotationen betrachten, die als semantische Annotationen immer als einfache RDF-Aussagetripel formuliert sind, und somit für die Endanwender nachvollziehbar sind. Bei den so verknüpften Ressourcen kann es sich nach dem Prinzip des *Linked (Open) Data* [BL06a] um beliebige Dokumente handeln, welche als Dokumentation der Datenbank dienen. Mit Hilfe von Suchanfragen lassen sich die Daten auf relevante Ausschnitte einschränken.

Manuelle Annotationen Zu Messfahrten oder einzelnen Elementen kann der Analyst nach Betrachtungen manuell Annotationen hinzufügen. Für diese Aufgabe benötigt er entsprechende Funktionen in der im letzten Abschnitt eingeführten Benutzerschnittstelle. Zum einen lassen sich so Verknüpfungen zu formalen Ressourcen einer Ontologie herstellen, welche eine klar definierte Aussage besitzen. Andererseits kann der Analyst Webseiten über ihre URL mit den Experimentaldaten verknüpfen und auf Herstellerangaben oder Lexikoneinträge verweisen. Existiert in der eigenen Organisationseinheit ein Wiki-System [SBBK07, Krö10], können auf diese Weise weitergehende Informationen zusätzlich zu den formalen Annotationen hinterlegt werden.

Semantische Technologien bieten eine Unterstützung für die mehrsprachige Arbeit mit Ressourcen (vgl. Abschnitt 2.3.1.2). Eine Unterstützung dieser Mehrsprachigkeit in der Bedienoberfläche ist erstrebenswert, um die Zusammenarbeit in internationalen Teams mit Annotationen zu fördern.

Serviceentwicklung Das Ergebnis der Serviceentwicklung durch einen Softwareingenieur ist ein neuer Service, der selbständig eine bestimmte Aufgabe der Datenanalyse übernimmt. Als Ergebnis der späteren Serviceausführung können neue Annotationen oder Datensätze in der Datenbank abgelegt werden. Die Annotationen können auch in den Algorithmen eines weiteren Services berücksichtigt werden, wobei diese über ein API abgerufen werden. Der fertiggestellte Service wird in einem Repository registriert. So kann der Service in dem Repository leicht gefunden und in eigene Auswertungen eingebunden werden.

Services orchestrieren und starten Die zur Auswertung von Datensätzen benötigten Services werden dem Analysten über eine Benutzerschnittstelle angeboten. Hier kann er die für sich relevanten Services auswählen und die notwendige Abarbeitungsreihenfolge festlegen. Somit *orchestriert* der Anwender die Services zu der gewünschten Zusammenstellung. Eine serverseitige Laufzeitumgebung für die Services sorgt für die Verarbeitung des zusammengestellten Arbeitsauftrags.

Experimente und automatisierte Services Im Regelbetrieb werden ständig neue Experimente durchgeführt, deren Messdaten in der Datenbank gespeichert werden. Um möglichst unmittelbar mit den Auswertungen beginnen zu können, werden direkt nach dem Datenbankimport automatisch vordefinierte Services für die neuen Datensätze ausgeführt. Dafür ist es notwendig, dass zuvor die Auswahl entsprechend dem Anwendungsfall „Services orchestrieren und starten“ erfolgt. So stehen ausgewählte Ergebnisse bereits möglichst zeitnah zur Verfügung.

Integration in Drittapplikationen Über die Datenbankschnittstellen können natürlich nicht nur die bisher in den Anwendungsfällen erwähnten Applikationen auf die Datensätze zugreifen, sondern beliebige Programme. Da die semantischen Annotationen jedoch in einem bestimmten Format in der Datenbank zusammen mit den relationalen Daten gespeichert sind, sollte nach Möglichkeit eine Abstraktionsschicht verwendet werden. Dank dieser Abstraktionsschicht muss nicht auf dem eigentlichen Datenmodell gearbeitet werden.

Zur Abstraktion können zum einen Software-APIs eingesetzt werden oder ganze Softwarearchitekturkonzepte wie *Open Services Gateway initiative* (OSGi, [OSG12, MLA10]), bei der Funktionalitäten über Module und Dienste bereitgestellt werden. Mögliche Beispielanwendungen zur weitergehenden Integration semantischer Annotationen sind Ærogator (Abschnitt 2.2.2.1) oder EAMole [Kös07, Kap. 3.6, S. 62 ff.]. In Ærogator bietet sich parallel zu den Zeitreihen die Darstellung der Annotationen zur manuellen Exploration an. Bei EAMole handelt es sich um eine Anwendung zum Data-Mining in Zeitreihen (vgl. Abschnitt 6.3).

3.2 Architektur zur Integration semantischer Technologien für Metadaten

Der letzte Abschnitt hat das neuartige Konzept semantischer Datenbank-Annotationen eingeführt. Um dieses theoretische Konzept in die Praxis umsetzen zu können, wird eine entsprechende Architektur für das zu realisierende System benötigt (vgl. [NBK09b]). Diese Architektur wird in diesem Abschnitt nach der Bottom-Up-Vorgehensweise eingeführt, sodass aufbauend auf zentralen Elementen schrittweise die Gesamtarchitektur abgeleitet wird. Grundsätzliche Ziele sind die Wiederverwendung existierender Methoden und Werkzeuge, die Skalierbarkeit und die Offenheit der entstehenden Lösung. Die Offenheit bezieht sich sowohl auf die technologische Ebene, sodass offene Standards eingesetzt werden sollen. Andererseits ist eine Offenheit

im Sinne des Semantic Web gemeint, dass eine reibungslose Integration mit anderen Wissensbasen möglich ist.

3.2.1 Umsetzung von Datenbank-Annotationen

Für die Annotation von Datenbank-Elementen wird zwischen Tupeln, Attributen, Relationen, Schemas oder Datenbanken unterschieden (vgl. Abb. 3-2). Mit dem gleichen Mechanismus lassen sich aber auch andere Datenbankkonzepte annotieren (Primärschlüssel, Fremdschlüssel, Constraints . . .). Dies wird allerdings an dieser Stelle nicht betrachtet, da sie für die diskutierten Szenarien zunächst irrelevant sind.

Es ist ein Mechanismus zu verwenden, der es ermöglicht, die oben angesprochenen Datenbank-Elemente in die RDF-Aussagen (Def. 4) mit einzubeziehen (vgl. [BL98]). Da RDF eindeutige Bezeichner für zu beschreibende Ressourcen benötigt, werden die zu integrierenden Datenbank-Elemente mittels entsprechender URIs identifiziert. Für die Betrachtungen in diesem Abschnitt werden der Einfachheit halber zunächst nur Tabellen bzw. Relationen berücksichtigt, die Behandlung von Datenbanken und Schemas erfolgt aber analog.

Basierend auf diesen Annahmen werden in Anlehnung an [BL98] die in Tab. 3-1 aufgeführten Konstruktionsvorschriften verwendet, um ein Datenbank-Element mit einer URI zu referenzieren. Relationen sind eindeutig über ihren Namen *table_name* bestimmt. Weiterhin seien den Attributen einer Tabelle eindeutig Namen *column_name* zugeordnet. Der Einfachheit halber sei angenommen, dass sowohl in *table_name* als auch in *column_name* der Schrägstrich (*/*) nicht erlaubt ist. Um die Tupel einer Relation eindeutig identifizieren zu können, muss die zu annotierende Relation einen *Unique Key* (eindeutigen Schlüssel) besitzen. Ohne Beschränkung der Allgemeinheit wird jeder Relation ein Attribut *row_id* hinzugefügt, welches je Relation eindeutige Ganzzahlen erhält und als *Surrogate Key* (Stellvertreterschlüssel) dient. Über die Kombination aus Tupel und Attribut können einzelne Attributausprägungen referenziert werden. Die in der Tabelle dargestellten URIs beginnen mit `http://www.dlr.de/ts` und beziehen sich somit beispielhaft auf den Namensraum des *Instituts für Verkehrssystemtechnik* (Transportation Systems, TS) am *Deutschen Zentrum für Luft- und Raumfahrt* (DLR).

Beispielsweise würde eine URI, die in der Datenbank `orcl` im Schema `U_MyUser` in der Relation `20100716_105523_expstate` die Attributausprägung des Attributs `Velocity` in dem Tupel mit dem Schlüssel 48 referenziert, wie folgt aussehen:
`http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/U_MYUSER/table/20100716_105523_expstate/column/VELOCITY/row/48`

Zurzeit ist eine Codierung des Datenbankservers für die Verwendung nicht notwendig, da lediglich ein Datenbankserver eingesetzt wird, ließe sich aber nach den gleichen Prinzipien leicht

Datenbank-Element	Transformation in URI
Datenbank	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*database_name*
Schema	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*database_name*/schema/*schema_name*
Relation (Tabelle)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*database_name*/schema/*schema_name*/ ↪ table/*table_name*
Attribut einer Relation (Spalte)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*database_name*/schema/*schema_name*/ ↪ table/*table_name*/column/*column_name*
Tupel einer Relation (Zeile)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*database_name*/schema/*schema_name*/ ↪ table/*table_name*/row/*row_id*
Attributausprägung einer Relation (Zelle)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*database_name*/schema/*schema_name*/ ↪ table/*table_name*/column/*column_name*/ ↪ row/*row_id*

Tabelle 3-1: Transformationsregeln von Datenbank-Elementen in URIs (frei nach [NBK11])

umsetzen. Dabei werden URIs erzeugt, die folgendem regulären Ausdruck entsprechen:

$$http://www.dlr.de/ts/dominion-datastore/db/[A-Za-z\_]+ \\ \hookrightarrow (/schema/[A-Za-z\_]+(/table/[A-Za-z\_]+ \\ \hookrightarrow (/column/[A-Za-z\_]+)?(/row/[A-Za-z\_]+)?)?)?$$

Das beschriebene Mapping-Verfahren ist eine Bijektion in den Zielraum, der durch den regulären Ausdruck bestimmt wird. So lässt sich das Mapping eindeutig invertieren und den URIs lassen sich leicht wieder ihre ursprünglichen Datenbank-Elemente zuordnen.

Mit der beschriebenen Vorgehensweise sind die Werte selbst von Attributausprägungen nicht in dem RDF-Modell abgelegt. Jedoch gibt es Arbeiten, die ein Mapping einer relationalen Datenbank in RDF ermöglichen (vgl. Abschnitt 2.3.6.3, z.B. [Biz03, BOCGP04]). So könnte man auf die Nutzdaten einer Datenbank mittels RDF zuzugreifen.

Ein solches generisches Mapping wird zurzeit jedoch explizit vermieden, da es einen deutlich größeren Umfang an Zeilen-/Zellen-Instanzen und RDF-Elementen zur Folge hätte. Diese werden jedoch nicht benötigt und würden unnötige Speicher- und Rechenleistungs-Ressourcen binden. Dies resultiert aus der in Abschnitt 1.1 eingeführten Tatsache, dass mit den Annotationen auf einer abstrakten Sichtweise gearbeitet wird, die auf den Messwerten aufbaut. Stattdessen werden daher für eine möglichst hohe Effizienz nur Datenbank-Elemente in URI-Ressourcen abgebildet und referenziert, die wirklich benötigt werden.

Abbildung 3-4 zeigt ein Beispiel, welches die konkrete Verwendung der eingeführten Annotationen verdeutlicht. In dem Beispiel wird wieder die kurz zuvor betrachtete Tabelle

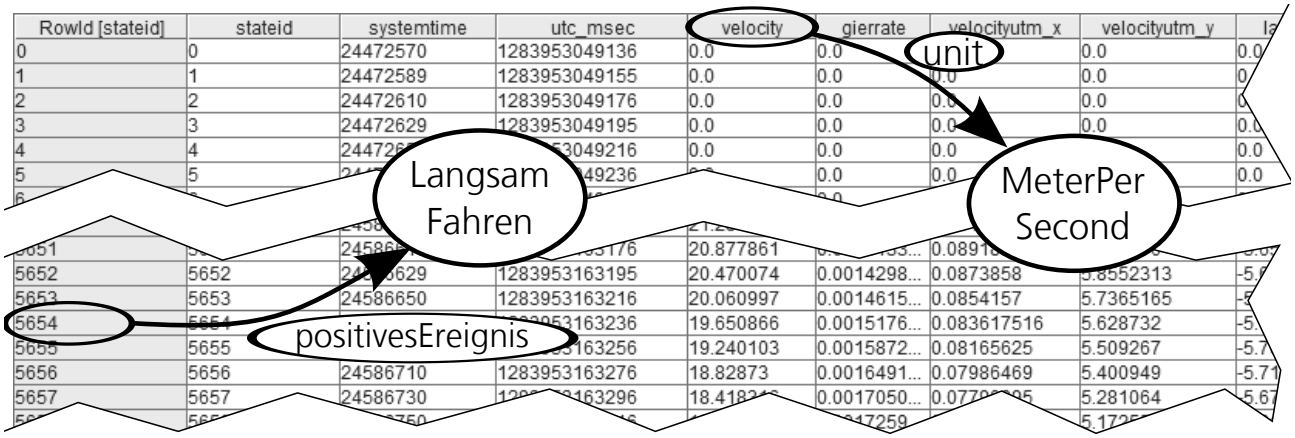


Abbildung 3-4: Beispiel von Annotationen[2] (frei nach [NBK09a])

`20100716_105523_expstate` herangezogen. In dieser sind zwei Annotationen dargestellt. Zum einen wird die Spalte `VELOCITY` über die Ontologie *Quantities, Units, Dimensions and Data Types in OWL and XML* (QUDT, [MHK10], vgl. Abschnitt 5.1) annotiert, dass die enthaltenen Werte in der physikalischen Einheit *Meter pro Sekunde* vorliegen. Als zweites wird die Zeile 5654 mit der Annotation versehen, dass das Ereignis *Langsam Fahren* detektiert wurde (vgl. Abschnitt 5.3). In Listing 3-1 sind diese in formaler Turtle-Notation abgebildet. In der Turtle-Notation wurden die Angaben zu den Namensräumen ergänzt, welche in einer Abbildung aus Gründen der Übersichtlichkeit nicht dargestellt werden können, aber erst zu einer Eindeutigkeit der getroffenen Aussagen führen.

```
@prefix driveEvent:    <http://www.dlr.de/ts/driveEvent.owl#> .
@prefix qudt:          <http://data.nasa.gov/qudt/owl/qudt#> .
@prefix unit:          <http://data.nasa.gov/qudt/owl/unit#> .

<http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/U_MYUSER
  ↪/table/20100716_105523_expstate/column/velocity>
    qudt:unit unit:MeterPerSecond .

<http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/U_MYUSER
  ↪/table/20100716_105523_expstate/row/5654>
    driveEvent:positivesEreignis driveEvent:Langsam_Fahren .
```

Listing 3-1: Annotationen aus Abb. 3-4 in Turtle-Notation

3.2.2 Details der Datenhaltung

Die zu den Massendaten anfallenden semantischen Annotationen müssen praktisch verwaltet werden. Die Umsetzung baut auf einer relationalen Datenbank auf, welche für die effiziente Verwaltung der Massendaten benötigt wird. Eine Speicherung der RDF-Annotationen in der Datenbank ist effizient möglich, wenn die Datenbank eine explizite Unterstützung für RDF bietet, z.B. [MAD+05]. Für die effiziente Speicherung von RDF in Datenbanken ergeben sich durch verschiedene Forschungsansätze neue Möglichkeiten, sodass in Zukunft noch weitere Fortschritte zu erwarten sind (vgl. Abschnitt 2.3.6.3). Eine Behandlung der RDF-Daten durch

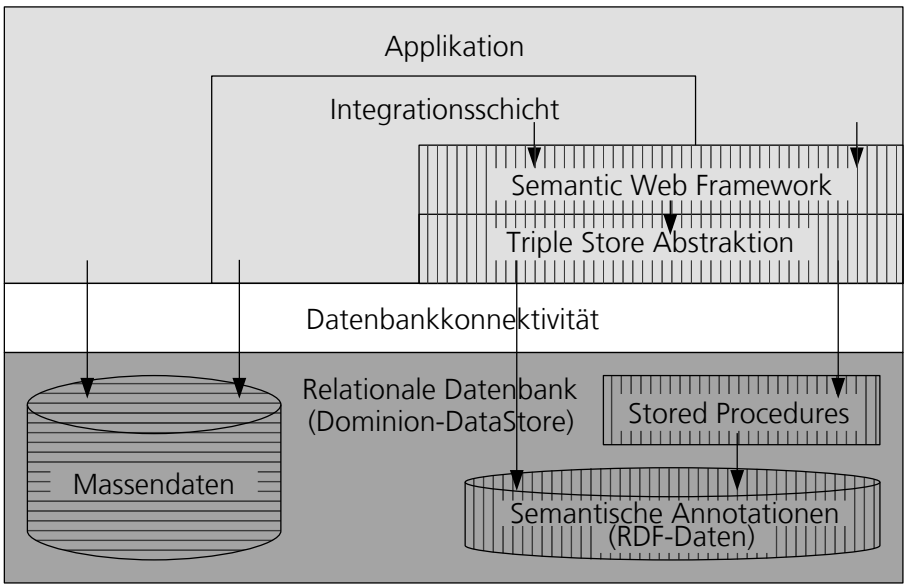


Abbildung 3-5: Übersicht der Architektur zum Datenmanagement[2] basierend auf Abb. 1-2 (nach [NBK11])

die Datenbank bringt weiterhin einen großen Vorteil mit sich. So kann über standardisierte Datenbankschnittstellen (SQL mit ODBC, JDBC) auf die RDF-Daten aus Anwendungen mit einer solchen Datenbankschnittstelle (z.B. Excel, R) zugegriffen werden (vgl. Abschnitt 4.3).

Als alternativer Ansatz könnte für die Speicherung der RDF-Daten eine spezialisierte RDF-Datenhaltung in Betracht gezogen werden, z.B. [AS08, BKvH02]. Eine solche bringt jedoch den Nachteil mit sich, dass die Massendaten und Annotationen nicht mehr über eine einheitliche Schnittstelle zugreifbar wären. Weiterhin bieten diese (im Gegensatz zu den Datenbanken) keine weitverbreitete und standardisierte Schnittstelle, welche durch Standardanwendungen (z.B. Excel, R) unterstützt werden würde. Eine gemeinsame Speicherung von Massendaten und Annotationen in einer RDF-Datenhaltung würde das Argument der getrennten Datenhaltung, nicht jedoch das Argument der nicht etablierten Schnittstellen entkräften. Weiterhin ist zu erwarten, dass eine vollständige Speicherung der Massendaten als RDF nicht effizient wäre (abhängig vom Umfang der Massendaten).

Unter der Voraussetzung, dass bestehende Technologien zur Datenhaltung eingesetzt werden, ist die gemeinsame Speicherung von Massendaten und RDF-Annotationen in einer Datenbank die vorteilhafteste Option. Weiterhin bringen relationale Datenbanken die üblichen Vorteile dieser ausgereiften Technologien mit sich (Mehrbenutzerfähigkeit, Transaktionen, Rechtemanagement . . .). Außerdem ist so eine Lösung aufbauend auf einer bereits bestehenden Haltung von Messdaten in einer Datenbank leicht möglich.

Aufbauend auf einer Datenbank ist in Abb. 3-5 die Architektur zur Datenspeicherung dargestellt, welche die Skizze aus der Einleitung in Abb. 1-2 soweit wie möglich basierend auf bereits existierenden Bausteinen umsetzt. In der untersten Ebene befindet sich die relationale Datenbank, die die Bezeichnung *Dominion-DataStore* trägt und welche sowohl die Massendaten als auch die RDF-Annotationen aufnimmt.

Diese beiden Datenbereiche werden in der gleichen Datenbank aber in unterschiedlichen, voneinander getrennten Strukturen vorgehalten. Sämtliche Elemente in der Grafik, die sich ausschließlich auf die Massendaten beziehen, sind horizontal schraffiert, während die Elemente mit Bezug zu den Annotationen vertikal markiert sind.

Die je Versuch aufgezeichneten Zeitreihen werden jeweils in einer separaten Tabelle abgelegt, sodass das anfallende Schema sehr einfach ist. Für jeden neuen Messpunkt fällt ein neuer Datensatz mit den erfassten Messwerten an. In der Datenbank sind außerdem *Stored Procedures* verfügbar, um die RDF-Daten auf SQL-Ebene direkt manipulieren zu können (Abschnitt 4.3). Die Datenbank an sich erlaubt somit den vollständigen Zugriff auf die Massendaten und deren semantische Annotationen. Auch wenn der Zugriff rudimentär ist, bietet die relationale Datenbank selbst eine grundlegende Abstraktionsebene, die bereits den vollständigen Zugriff auf alle Daten in einer einheitlichen Form ermöglicht. Die Verbindung zur Datenbank wird über eine übliche Datenbankzugriffsschicht hergestellt (z.B. JDBC, ODBC, Herstellerbibliothek). Auf die Massendaten können die Applikationen direkt zugreifen oder Hilfsfunktionen aus der Integrationsschicht benutzen.

Der Zugriff auf die RDF-Annotationen ist stärker abstrahiert, um das Datenmodell möglichst effizient behandeln zu können. Die praktische Umsetzung der Speicherung erlaubt mehrere Alternativen, welche neben den Stored Procedures durch die *Triple Store Abstraktion* umgesetzt wird (vgl. Abschnitt 2.3.6.3). Vorzugsweise wird derzeit die native Möglichkeit zur RDF-Speicherung der eingesetzten Oracle-Datenbank verwendet, da diese am effizientesten erscheint. Alternativ kann ein von dem Datenbanksystem unabhängiges Schemalayout zur Speicherung der Annotationen eingesetzt werden (vgl.a. [Vel10]).

Auf die Speicherabstraktion setzt ein übliches *Semantic Web Framework* (z.B. [Sou08, CDD+04]) auf, welches die in den Speicher geladenen Annotationsaussagen in Form eines Graphen verwaltet (vgl. Abschnitt 3.4). Je nach eingesetztem Graphen, kann es sich lediglich um RDF-Tripel handeln oder darauf aufbauend um ein RDFS- oder OWL-Modell.

Weiterhin existiert die neuartige *Integrationsschicht*, welche die Datenbank-Elemente mit den Semantic Web Technologien verbindet (vgl. Abschnitt 4.1). Sie erlaubt, die Datenbank-Elemente mit ihren Ressourcen-URIs zu verknüpfen, sodass sich RDF-Tripel den Datenbank-Elementen zuordnen lassen. Die softwaretechnische Umsetzung basiert auf den in Abschnitt 3.2.1 vorgestellten Prinzipien. Die RDF-Speicherung erlaubt die Verwaltung verschiedener Modelle, welche je eine Menge von Aussagen umfassen und untereinander verlinkt sein können (vgl. Abschnitt 3.2.3). Die Verwaltung dieser Modelle erfolgt ebenfalls durch die Integrationsschicht.

Die strikte Trennung der Massendaten und der Annotationen spiegelt deren unterschiedliche Verwendung wider. Messdaten sind im Nachhinein keinen Änderungen mehr unterworfen und werden als statisch betrachtet. Ihre Aussagen bestehen in den in Zahlen codierten Messwerten, welche interpretiert werden. Über dieser Betrachtungsebene liegt die abstrakte Sichtweise, welche auf Interpretationen beruht. Auf dieser Ebene sind Beschreibungen (z.B. Spaltenbeschrei-

bungen) oder identifizierte Ereignisse im zeitlichen Ablauf (z.B. langsames Fahren) zu sehen. Bei den so getroffenen Aussagen handelt es sich normalerweise um assertionales Wissen (ABox, vgl. Abschnitt 2.3.3.4). Das Schemawissen (TBox) kann wie bei RDFS/OWL üblich mit einem Editor wie Protégé durch Anwender bearbeitet und eingebunden werden (vgl. Abschnitt 4.2).

Die strikte Trennung bietet zusätzliche Chancen. Werden beispielsweise annotierte Massendaten im Laufe der Zeit aus der Datenbank entfernt, weil sie zu viel Platz benötigen und daher archiviert werden, können die Annotationen in der Datenbank weiterexistieren. Ein solcher RDF-Index kann somit weiterhin nachträglich durchsucht werden, sodass relevante Tabellen bei Bedarf gezielt aus dem Archiv zurückgeholt werden können.

3.2.3 Verteilung der RDF-Daten auf mehrere Modelle

Ein wesentlicher Schritt bei der Behandlung semantischer Annotationen ist deren Verwaltung, um eine geeignete Skalierbarkeit zu erreichen. Würden sämtliche Aussagen für alle durchgeführten Experimente (z.B. Messfahrten) bzw. Tabellen zusammen in einem Graph-Modell (Def. 1 – 3) verwaltet, kann das Wachstum der Wissensbasis als ungefähr proportional zur Anzahl der Experimente / Tabellen angenommen werden. Dieser Aussage liegt die Annahme zu Grunde, dass sich die meisten Aussagen auf bestimmte Tabellen oder deren Inhalte beziehen, da diese die wesentlichen Informationen, die eigentlichen Nutzdaten, enthalten. Ist irgendwann der Zeitpunkt erreicht, dass Operationen (z.B. Such-, SPARQL-Anfrage, Reasoning) auf dem angenommenen gemeinsamen Graph-Modell zu lange dauern, lässt sich keines der Experimente mehr in angemessener Zeit betrachten (vgl. Abschnitt 5.3).

Aus diesem Grund müssen die Annotationen auf verschiedene Graph-Modelle aufgeteilt werden, wobei das Ziel ist, eine Skalierbarkeit auf der Ebene einzelner Tabellen zu erreichen. Eine Übersicht über die Aufteilung in die einzelnen Graph-Modelle ist in Abb. 3-6 dargestellt. Gearbeitet wird grundsätzlich auf dem *Basis-Graph-Modell*, welches als Graph G_B bezeichnet wird. Dieser Graph stellt die Vereinigung der tatsächlich verwendeten Teilgraphen dar. Je nach Bedarf werden dynamisch Teilgraphen hinzugefügt oder wieder entfernt, was dem Anwender somit verborgen bleibt. Dabei sind in Abb. 3-6 konkrete Teilgraphen mit eigenständigen Aussagen die Blätter des dargestellten Baumes und durch eine durchgezogene Linie gekennzeichnet. Die inneren Knoten (gestrichelt umrandet) stellen somit hierarchisierende Elemente dar, welche die Aussagen der Kindgraphen enthalten, aber keine eigenen.

Auf der linken Seite der Abbildung ist das Datenbank-Graph-Modell G_D dargestellt, welches die Annotationen bezüglich der betrachteten Datenbank-Elemente enthält. Diese werden nach Abb. 3-5 durch die Integrationsschicht verwaltet und im Bereich für *RDF-Daten* der Datenbank gespeichert. G_D umfasst zum einen die Schema-Annotationen G_S, welche sich auf eine Datenbank `D` (Graph G_{SD_D}), ein Schema `S` ($G_{SS\_D/S}$) oder eine Tabelle `T` ($G_{ST\_D/S/T}$, inklusive Spalten) beziehen. Diese Elemente werden jeweils nur auf Schema-Ebene betrachtet, also ohne

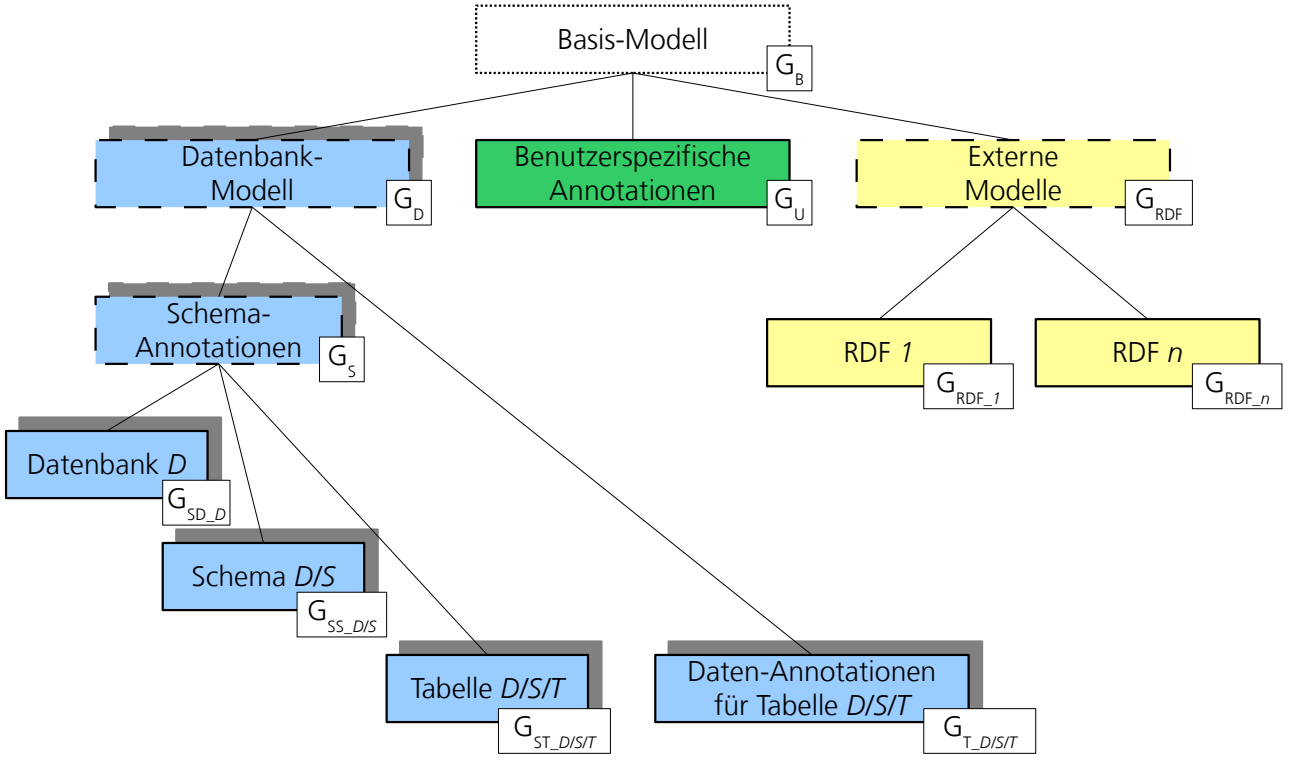


Abbildung 3-6: Verteilung der Daten auf unterschiedliche RDF-Modelle

die in der Tabelle enthaltenen Datensätze und Werte. Die eigentlichen (Mikro-)Annotationen auf Ebene der Zeilen oder Zellen sind im Graphen $G_{T\_D/S/T}$ enthalten. Somit gilt:

$$G_D = G_S \cup G_{T\_D/S/T} = (G_{SD\_D} \cup G_{SS\_D/S} \cup G_{ST\_D/S/T}) \cup G_{T\_D/S/T}$$

Es existiert für jede betrachtete Instanz dieser Datenbank-Elemente ein separates Graph-Modell. Die Tatsache, dass es jeweils mehr als nur ein Graph-Modell für die Teilgraphen von G_D gibt, wird in der Abbildung durch Schatten dargestellt. Wird die Beispieltabelle `20100716_105523_expstate` im Schema `U_MyUser` in der Datenbank `orcl` aus Abb. 3-4 betrachtet, lauten die Namen der konkreten zu betrachtenden Graphen wie folgt: G_{SD_orcl}, G_{SS_orcl/U_MYUSER}, $G_{ST\_orcl/U\_MYUSER/20100716\_105523\_expstate}$ und $G_{T\_orcl/U\_MYUSER/20100716\_105523\_expstate}$.

Das Graph-Modell G_U enthält nutzerspezifische Annotationen, welche im Gegensatz zu G_D nur für den aktuell arbeitenden Datenbankbenutzer verfügbar sind. In ihm können Annotationen und Aussagen zu Testzwecken abgelegt werden. Sollten umfangreichere Mengen von Annotationen gespeichert werden, wäre eine Aufteilung in Teilgraphenmodelle analog zu G_D möglich und notwendig.

Häufig ist es notwendig, externe Wissensbasen G_{RDF_i} in Form von RDF-Dokumenten zusätzlich mit zu berücksichtigen, ohne sie mit in die Datenbank übernehmen zu müssen. Diese können z.B. Ontologien im OWL-Format enthalten. Hierfür sei $G_{RDF} = \bigcup_{i \in I} G_{RDF\_i}$ mit $i \in I$ die Menge der relevanten RDF-Dokumente. Somit können die G_{RDF_i} ebenfalls G_B als Teilgraph

hinzugefügt werden, sodass gilt:

$$G_B = G_D \cup G_U \cup G_{RDF}$$

Ein RDF-Modell kann aus einer lokalen Datei, aus dem Internet per HTTP, aus einem semantischen Wiki oder als separates Modell (ohne Bezug zu G_D) aus einer Datenbank geladen werden. Im vorliegenden Anwendungsfall der Messdatenverarbeitung enthält ein zusätzliches RDF/OWL-Dokument eher Schemawissen (TBox), während assertionales Wissen (ABox) bezüglich der Datenbank-Elemente in G_D enthalten ist.

Sei der Betrag $|G|$ eines Graphen G die Summe aus Anzahl Knoten $|V(G)|$ und Anzahl Kanten $|E(G)|$. Dann sei $|G|$ im Folgenden eine näherungsweise Maßzahl für den Aufwand von komplexen Operationen (z.B. SPARQL, Reasoning) auf G, sodass anhand des Betrages zwei Graphen miteinander verglichen werden können. Tatsächlich hängt der Aufwand einer Operation noch von dem konkreten Aufbau der Anfrage (Struktur SPARQL-Query, Reasoning-Regeln) und der individuellen Struktur des Graphen ab. Für eine erste Abschätzung sei aber $|G|$ als ausreichend betrachtet.

Sei $G = G_1 \cup G_2$, dann ist $|G| \leq |G_1| + |G_2|$. Die Gleichheit gilt genau für den Fall, dass die Mengen der Knoten und Kanten jeweils disjunkt sind. Für die Vereinigung der Graphen in Abb. 3-6 kann dies aufgrund der entsprechenden Konstruktion der Teilgraphenmodelle zumindest für die Kanten und als Näherung für die Knoten angenommen werden, sodass gilt:

$$|G_B| \approx |G_D| + |G_U| + |G_{RDF}|$$

Zur weiteren Abschätzung sei $G = G_1 \cup G_2$ und $|G_2| \ll |G_1|$, dann ist der Einfluss von G_2 vernachlässigbar, sodass $|G| \approx |G_1|$. Da G_U von der Verwendung so konstruiert ist, dass es nur eine geringe Menge an Aussagen im Vergleich zu G_D enthält, und somit $|G_U| \ll |G_D|$ gilt, folgt daraus:

$$|G_B| \approx |G_D| + |G_{RDF}| \approx |G_S| + |G_{T\_D/S/T}| + |G_{RDF}|$$

Weiterhin wurde zu Beginn dieses Abschnitts eingeführt, dass der Großteil der Annotationen von G_D sich in den $G_{T\_D/S/T}$ befindet, da dort die Annotationen für die eigentlichen Nutzdaten gespeichert werden. Somit gilt $|G_S| \ll |G_{T\_D/S/T}|$ und daraus folgt:

$$|G_B| \approx |G_{T\_D/S/T}| + |G_{RDF}|$$

Nach dieser Näherung fallen also bei der Betrachtung semantischer Annotationen insbesondere die Annotationen der Nutzdaten ($G_{T\_D/S/T}$) und die zusätzlich eingebundenen externen Wissensbasen ins Gewicht. G_{RDF} kann sowohl trivial bis hin zu sehr komplex sein, da es vollständig unter der Kontrolle des Anwenders liegt, weswegen auch keine weiteren Abschätzungen möglich sind. Der Anwender ist somit selbst dafür verantwortlich, nur solche externen Wissensbasen

mit zu betrachten, dass die durchzuführenden Betrachtungen vom Aufwand her noch möglich sind.

Aus der letzten Gleichung ergibt sich die angezielte Skalierbarkeit auf Tabellen-Ebene. Ein Graph G_B zu einer betrachteten Tabelle T wird maßgeblich durch die Daten-Annotationen dieser Tabelle ($G_{T\_D/S/T}$) und den Graphen G_{RDF} bestimmt. Sind gewünschte Betrachtungen zu einer Tabelle T auf einem Graph G_B möglich, sind die gleichen Betrachtungen möglich für eine Tabelle T', wenn $|G_{T\_D/S/T}| \approx |G_{T\_D/S/T'}|$ gilt. Die Tatsache, dass Annotationen für beliebig viele weitere Tabellen in der Datenbank vorliegen, ist für die Betrachtung irrelevant. Da immer nur die benötigten Teilgraphen von G_B instanziiert werden, gelten die zuvor durchgeführten Komplexitätsbetrachtungen. Können gewünschte Betrachtungen für eine Tabelle durchgeführt werden, können somit beliebig viele Tabellen für semantische Annotationen erfolgreich eingesetzt werden.

Die in Abb. 3-6 durchgeführte Aufteilung der Schema-Annotationen G_S in weitere Teilgraphen, ist für die soeben durchgeführten Betrachtungen nicht zwingend notwendig gewesen. Diese Aufteilung erhöht jedoch (bei einer Implementierung) die Zukunftssicherheit für den Fall, dass in Zukunft irgendwann sehr viele Tabellen vorliegen. Würden nämlich die Tabellen eine ähnliche Größenordnung wie die Nutzdaten erreichen, würden die Schema-Annotationen G_S in die gleichen Dimensionen wie die Annotationen der Nutzdaten wachsen. Bei den betrachteten Messwerten (vgl. Abschnitt 5.3) fallen bei einer Abtastrate von 10 ms in einer Stunde 360.000 Datensätze bzw. Zeilen an, wogegen sich die Anzahl der Experimente bzw. Tabellen üblicherweise eher im zweistelligen Bereich bewegt (Erfahrungswert). Sollten tatsächlich zu viele Schema-Annotationen in G_S vorliegen, werden ebenfalls nur die notwendigen Teilgraphen instanziiert, welche notwendig sind, die relevante Tabelle zu betrachten.

Für die Blätter des Graph-Modells G_D aus Abb. 3-6, welche die eigentlichen Datenbank-Element-Annotationen enthalten, werden bei der Speicherung Namen in Form von URIs vergeben. Diese orientieren sich an den Namen aus Tab. 3-1 und lauten wie in Tab. 3-2 dargestellt. Die URIs für die Modelle erfüllen eine ähnliche Funktion wie die Namen von Tabellen in relationalen Datenbanken und dienen zur Referenzierung der Modelle. Beispielsweise können Graphen in SPARQL-Abfragen über das `FROM`-Schlüsselwort direkt referenziert werden (vgl. [PS08, Kap. 8.2]), wenn dies durch die Query-Engine unterstützt wird.

Damit Annotationen immer im passenden Modell vorliegen, sodass sie verfügbar sind, wenn sie benötigt werden, ergeben sich neuartige Regeln zur Speicherung für semantische Annotationen (nach Def. 4). Diese sind für jeden Annotationstripel der Reihe nach abzuarbeiten und die Verarbeitung ist abzubrechen, sobald eine Regel erfolgreich angewendet werden konnte:

1. Wenn die URI der Datenbank D Subjekt oder Objekt der Annotation ist, dann speichere die Annotation in G_{SD_D}.
2. Wenn die URI des Schemas D/S Subjekt oder Objekt der Annotation ist, dann speichere die Annotation in $G_{SS\_D/S}$.

Graph-Modell	Transformation in URI
G_{SD_D} (Datenbank)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*D*#model
$G_{SS\_D/S}$ (Schema)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*D*/schema/*S*#model
$G_{ST\_D/S/T}$ (Tabelle)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*D*/schema/*S*/table/*T*#model
$G_{T\_D/S/T}$ (Tabelleninhalte)	http://www.dlr.de/ts/dominion-datastore/ ↪ db/*D*/schema/*S*/table/*T*#bulk-model

Tabelle 3-2: Regeln zur Erzeugung von URIs für Graph-Modelle

3. Wenn die URI der Tabelle D/S/T oder eine ihrer Spalten das Subjekt oder Objekt der Annotation ist, dann speichere die Annotation in $G_{ST\_D/S/T}$.
4. Wenn die URI einer Zeile oder Zelle der Tabelle D/S/T das Subjekt oder Objekt der Annotation ist, dann speichere die Annotation in $G_{T\_D/S/T}$.

3.2.4 Komponenten im verteilten Informationssystem

Nach Einführung der Prinzipien semantischer Annotationen, ihrer Umsetzung und Verwaltung werden die Annotationen in diesem Abschnitt als Teil eines verteilten Systems betrachtet, damit sie durch dessen Softwarekomponenten benutzt werden können. Es wird gezeigt, wie in einer solchen verteilten Architektur verschiedene Komponenten miteinander interagieren können. Zentrum der Sichtweise in diesem Abschnitt sind die Datenbank und ihre Annotationen. Die einzelnen Komponenten sind weitgehend sehr lose über offene (Web-) Standards miteinander gekoppelt, sodass eine Umsetzung nach den Entwurfsprinzipien einer *serviceorientierten Architektur* (SOA, [Jos08]) vorliegt. Somit übernehmen die einzelnen Komponenten jeweils eine separate, klar definierte Aufgabe und bieten ihren *Dienst* zur Weiterverwendung an. Diese können je nach Bedarf kombiniert bzw. orchestriert werden, um andere komplexere Aufgaben zu erfüllen. Es wird möglichst umfassend dargestellt, wie unterschiedliche Komponenten sinnvoll miteinander kombiniert werden können. Da zur Kopplung jedoch offene Standards eingesetzt werden, lassen sich auch nicht aufgeführte Systeme bei Bedarf anbinden. Außerdem werden im Folgenden lediglich Komponentenklassen betrachtet, sodass eine konkrete Ausgestaltung der Komponenten nach individuellen Bedürfnissen weiterhin möglich ist.

Eine Übersicht über die Komponenten, deren Interaktion im Folgenden beschrieben wird, ist in Abb. 3-7 dargestellt. Im Zentrum ist der Datenbankserver mit den Massendaten und den semantischen Annotationen in Form von RDF-Aussagen abgebildet. Die einzelnen Rechtecke stellen Komponenten dar, welche in Bezug mit semantisch annotierten Daten der Datenbank stehen. Der Bezug kann sowohl von der Komponente ausgehen, dass sie auf die Daten zugreift oder sie referenziert. Andererseits können die semantischen Annotationen einen Bezug zu fremden Komponenten besitzen.

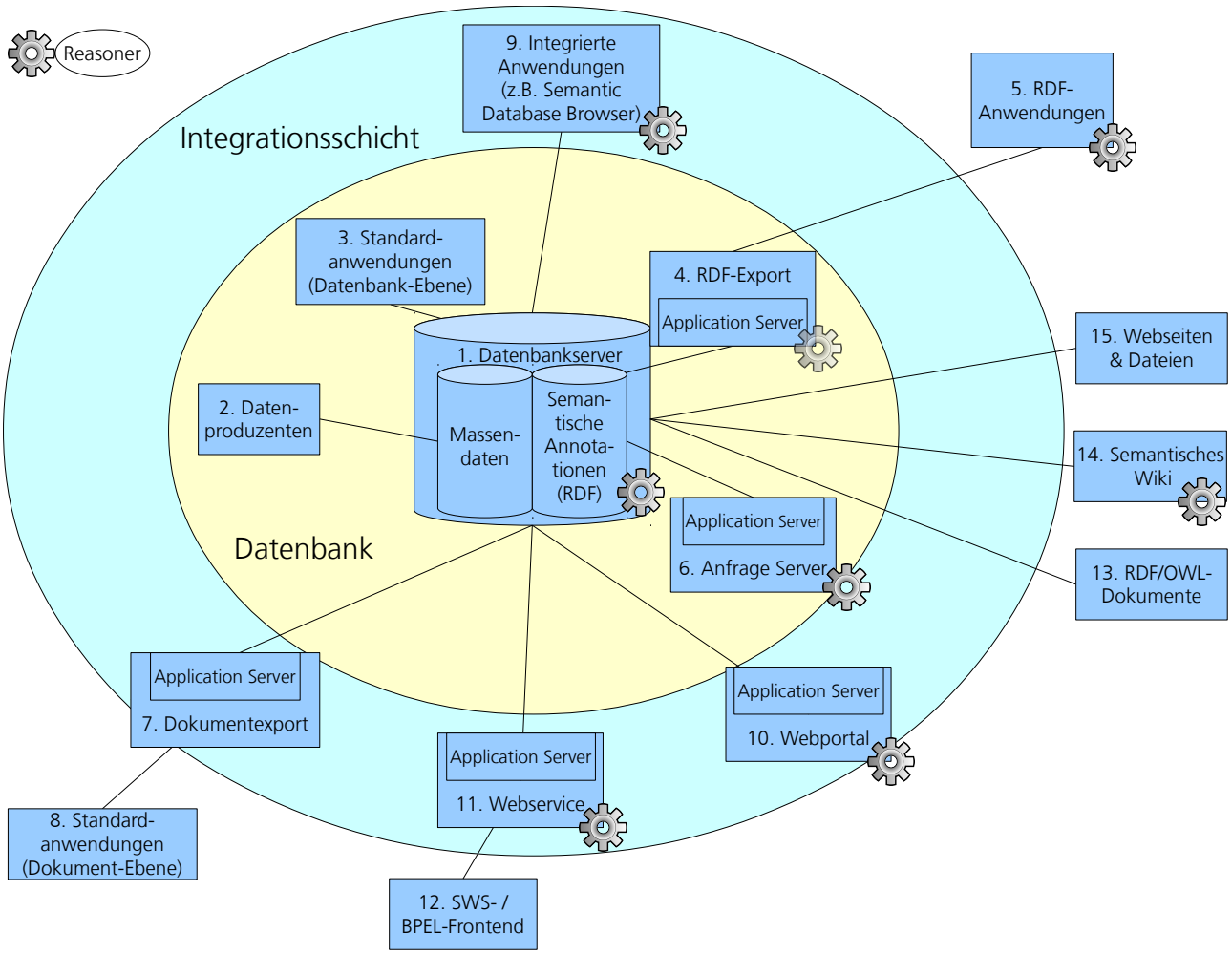


Abbildung 3-7: Semantische Annotationen im Kontext eines verteilten Systems

Es werden für den Zugriff auf die semantisch annotierten Daten drei Abstraktionsebenen unterschieden, welche in der Abbildung durch die großen Ovale dargestellt sind. Komponenten, die im inneren Oval dargestellt sind, greifen über standardisierte Datenbankschnittstellen auf die Daten zu (vgl. Abschnitt 4.3). Da sie so auf die Funktionalitäten der Integrationsschicht der Architektur aus Abb. 3-5 verzichten müssen, besitzen sie nur eine bedingt integrierte Sicht auf die Daten und ihre Annotationen. Im nächst größeren Oval sind die Komponenten angeordnet, welche über die gemeinsame Integrationsschicht zugreifen und somit eine integrierte Sicht auf die Daten und deren Annotationen besitzen. Außerhalb der Ovale sind Elemente dargestellt, die auf eine sonstige Art und Weise in Bezug zu den semantisch annotierten Daten stehen. Zum einen können sie in einem logischen Zusammenhang stehen, weil sie durch RDF-Aussagen verknüpft sind, oder sie greifen lediglich mittelbar über eine weitere Komponente auf die Daten zu.

Für einige Komponenten ist weiterhin in der Abbildung vermerkt, dass es sich anbietet, diese durch einen Application-Server ausführen zu lassen, da dieser diverse Dienste zur Wiederverwendung anbietet. Einige Komponenten sind außerdem mit einem Zahnrad markiert, um darzustellen, dass diese einen Reasoner einsetzen können, um die semantischen Aussagen zu interpretieren.

1. **Datenbankserver** Der *Datenbankserver* entspricht der unteren Schicht aus Abb. 3-5 und enthält die *Massendaten* und ihre *Annotationen* in logisch getrennten Einheiten. Datenbanken mit nativer Unterstützung semantischer Technologien bieten ggf. direkt eine Reasoner-Unterstützung für die gespeicherten Annotationen.
2. **Datenproduzenten** Sie produzieren in großem Umfang tabellarische Massendaten. Es handelt sich z.B. um Experimental- oder Sensorsysteme. Im Folgenden wird für die Umsetzung davon ausgegangen, dass die konkreten Experimentalsysteme auf der *Dominion*-Plattform aufbauen (vgl. Abschnitt 3.2.5) und somit die Datenbank für sämtliche darauf aufbauende Systeme zur Verfügung steht.
3. **Standardanwendungen (Datenbank-Ebene)** Unter dieser Kategorie werden Anwendungen zusammengefasst, welche über standardisierte Datenbankschnittstellen auf die Daten zugreifen (z.B. Excel, R, DIAdem, SPSS). Mit den in Abschnitt 4.3 aufgeführten Methoden können zugleich die semantischen Annotationen mit berücksichtigt werden.
4. **RDF-Export** Der *RDF-Export* stellt die gespeicherten semantischen Annotationen über HTTP als RDF-Dokumente zur Verfügung. Diese können (z.B.) über die URIs der Modelle (Abschnitt 3.2.3) abgerufen werden, wobei die Dokumente dynamisch erzeugt werden und somit immer aktuell sind. Der in einem Application-Server ausgeführte Export greift über ein Semantic Web Framework (vgl. Abb. 3-5) auf die Annotationen zu.
5. **RDF-Anwendungen** Unter diesem Sammelbegriff werden Applikationen zur Verarbeitung und Darstellung von RDF- und OWL-Wissensbasen zusammengefasst (z.B. Protégé, vgl. Abschnitt 2.3.6). Diese Repräsentationen werden i.d.R. direkt als *RDF/OWL-Dokument* geöffnet oder aber dynamisch von einer Server-Anwendung (z.B. *RDF-Export*, *semantisches Wiki*) erzeugt.
6. **Anfrageserver** Ein *RDF-Anfrageserver* erlaubt es, Abfragen ähnlich zu Datenbankabfragen auf RDF-Wissensbasen durchzuführen, und bietet dieses Angebot als separaten Dienst an (z.B. [Jen11, CDD+04, AS08, BKvH02]). Als Abfragesprache wird zumindest SPARQL angeboten. Der Zugriff erfolgt über ein Netzwerkprotokoll (z.B. [CFT08]) oder über eine Webseite. Auf die semantischen Annotationen wird direkt über ein Semantic Web Framework zugegriffen (Abb. 3-5). Die Umsetzung innerhalb eines Application-Servers ist möglich.
7. **Dokumentexport** Der *Dokumentexport* ermöglicht es, Massendaten in Kombination mit Annotationen in eine separate Datei zu exportieren. Das logische Format orientiert sich an den Tabellen der Massendaten, wobei zusätzliche Spalten hinzugefügt werden, welche die Annotationen beinhalten (vgl. Abschnitt 4.2 und 4.3). Eine physische Speicherung kann beispielsweise im CSV-, Excel-, XML- oder HTML-Format erfolgen. Das logische Format stellt ein Hilfskonstrukt dar, um auf Tabellen arbeitenden Anwendungen (z.B. Excel) eine grundlegende Verarbeitung der Annotationen zu ermöglichen.
8. **Standardanwendung (Dokument-Ebene)** Diese Kategorie fasst Anwendungen zusammen, welche als Tabellen strukturierte Dateien verarbeiten (z.B.Excel, DIAdem). Parallel zu den Massendaten wird auf die semantischen Annotationen zugegriffen, indem sie als durch den *Dokumentexport* bereitgestellte, tabellenbasierte Dateien verarbeitet werden.

9. **Integrierte Anwendungen** Sie sind auf die Verarbeitung von Massendaten in Kombination mit semantischen Annotationen ausgelegt (z.B. Semantic Database Browser, Abschnitt 4.2). Daher setzen sie vorzugsweise auf die Integrationsschicht der Architektur (Abb. 3-5) auf, um von der dort bereitgestellten Funktionalität zu profitieren, und laufen als lokale Applikation beim Anwender. In diese Kategorie fallen auch *Informationssysteme für Fahrversuche* (Abschnitt 2.2.2.1).
10. **Webportale** Diese werden über einen Internet Browser bedient. Die technische Umsetzung erfolgt aufbauend auf die Integrationsschicht der Architektur und innerhalb eines Web- bzw. Application-Servers. Auf diese Weise können Portale ähnlich wie in *wissenschaftlichen Datenhaltungssystemen* (Abschnitt 2.2.4) realisiert werden. Semantische Annotationen erlauben neue Möglichkeiten durch die Verwendung von Template- und Mashup-Techniken, um fremde Informationsquellen dynamisch zur Anzeige mit aufzubereiten (vgl. [ADD10]).
11. **Webservices** Sie sind Dienste, die ohne weitere Benutzerinteraktion Berechnungen (z.B. Messdatenanalyse) auf der Datenbank durchführen (vgl. Abschnitt 3.1.2). Sie können semantische Annotationen verarbeiten oder neue erzeugen. Zum Zugriff wird daher die Integrationsschicht der Architektur verwendet. Die Implementierung erfolgt in einem Application-Server, da über diesen die Schnittstellen zur Ansteuerung des Webservices bereitgestellt werden.
12. **SWS/BPEL-Frontend** Die Zusammenstellung und Ausführung von *Webservices* mittels Ablaufplänen erfolgt durch *SWS/BPEL-Frontends* (vgl. Abschnitt 2.2.3, 3.1.2). Ist ein Ablaufplan durch einen Anwender erstellt, kann dieser auf einem separaten Server zur Ausführung abgelegt werden.
13. **RDF/OWL-Dokumente** Sie stellen eine separate Wissensbasis dar, welche in einem logischen Bezug zu den semantischen Annotationen und somit zu den Massendaten steht. Es existiert folglich ein RDF-Aussagetripel, welches zugleich das Dokument und ein Datenbank-Element enthält. Je nachdem ob dieses Tripel Element der Annotationen oder des Dokumentes ist, geht der Bezug von den Annotationen oder vom Dokument aus.
14. **Semantisches Wiki** Ein *semantisches Wiki* ist eine Spezialform der *RDF/OWL-Dokumente*, die die Vorteile von RDF-Aussagen mit den Funktionen eines klassischen Wikis (optisch aufbereiteter Fließtext, Archivierung) verknüpft (vgl. [SBBK07, Krö10]). Parallel zu den textuellen Beiträgen eines Wikis werden Aussagen in formaler Form als RDF-Tripel vorgehalten.
15. **Webseiten & Dateien** Diese werden in Annotationen über ihre URI referenziert, wobei der Bezug von den Annotationen ausgeht. Die Interpretation eines solchen Tripels bzw. der URI einer Webseite oder Datei obliegt dem Nutzer (vgl. Abschnitt 2.3). Jedoch verweisen einige Prädikate (z.B. `rdfs:seeAlso` [BG04, Kap. 5.4.1] und Annotation-Properties [MPSP09, Kap. 5.5]) explizit auf Dokumente.

Die gezeigte, neuartige Verknüpfung von Komponenten aus der Welt der relationalen Datenbanken mit Semantic Web Werkzeugen in einem gemeinsamen verteilten System erlaubt somit deren gemeinsame Benutzung zur Arbeit auf der gleichen Datenbasis. Auf diese Weise werden jedoch nicht nur Werkzeuge miteinander verknüpft, sondern auch Sichtweisen und Methoden

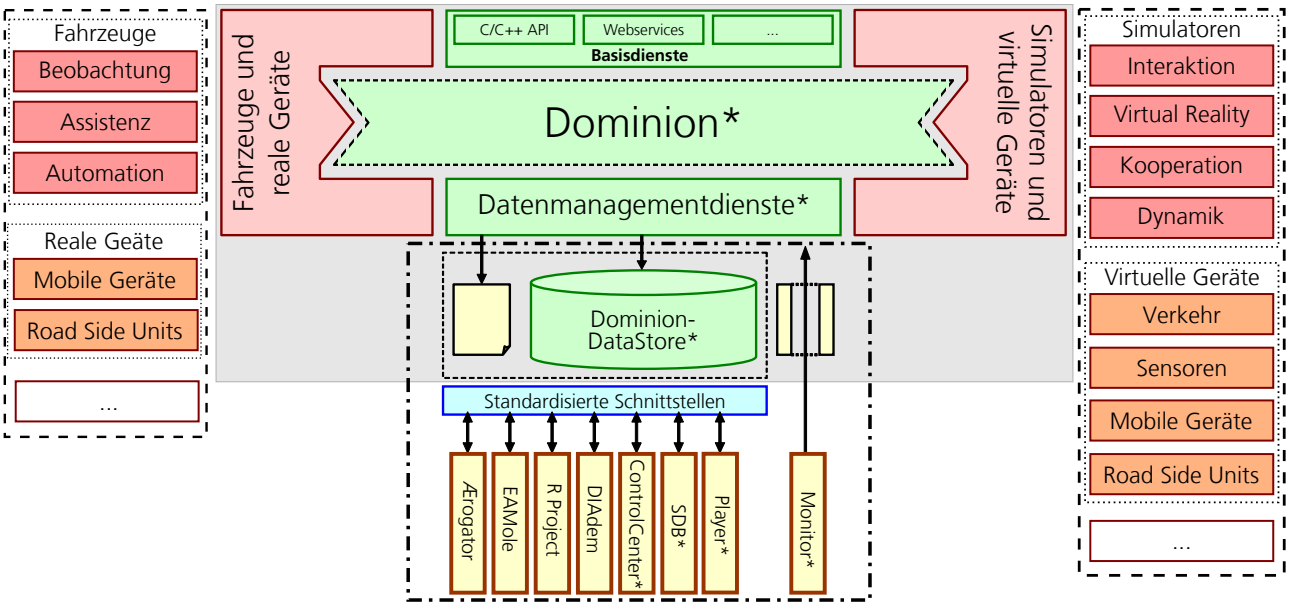


Abbildung 3-8: Datenzugriff durch Analyseanwendungen auf *Dominion*-Experimente[2] (frei nach [SHG+10])

aus zwei unterschiedlichen Bereichen. Ziel dieser Betrachtungen ist die Verarbeitung von Messdaten, die, wie im nächsten Kapitel gezeigt, erhoben und eingebunden werden können.

3.2.5 Anbindung an Experimentalsysteme

Für die Durchführung von Experimenten und Versuchen ergibt sich für einen Anwender die entscheidende Frage nach dem Zugriff der Daten aus den Experimentalsystemen. Die logische Sicht dieser Perspektive und deren Umsetzung sind in Abb. 3-8 dargestellt. *Dominion* stellt somit eine Realisierung der Komponente *Datenproduzenten* aus Abb. 3-7 dar, während der *Dominion-DataStore* dem *Datenbankserver* entspricht. Die mit einem Stern (*) markierten Komponenten sind im Rahmen dieser Arbeit entstanden oder wurden wesentlich weiterentwickelt.

Im Zentrum von Abb. 3-8 befindet sich die gemeinsame Architekturplattform bzw. Middleware *Dominion*. Sie abstrahiert den Betrieb verschiedener Experimentalsysteme in Form von verschiedenen Fahrzeugen, Simulatoren und weiteren Geräten. Dank *Dominion* kann auf einer einheitlichen, logischen Ebene zur Darstellung der Variablen und Messwerte für alle beteiligten Experimentkomponenten gearbeitet werden (vgl. [MDD+10, GHHK08]). Diese zentrale, logische Darstellung der Variablen wird in der *Dominion*-Terminologie als Datenkern bezeichnet. Die Architekturplattform enthält neben der technischen Umsetzung als Software ebenfalls Unterstützung für die Entwicklung und Betrieb der Experimentalsysteme. Zur Unterstützung werden Entwicklungs- und Programmierwerkzeuge, Methoden und Prozesse angeboten (vgl.a. [SHG+10, SHN+11]). Über die *Datenmanagementdienste* werden die Simulationsdaten als Zeitreihen aufgezeichnet und im Dominion-DataStore, einer relationalen Datenbank, gespeichert. Die Anbindung des Dominion-DataStore wird direkt durch Komponenten in *Dominion* realisiert, sodass sämtliche auf *Dominion* implementierte Systeme eine Anbindung an den Dominion-DataStore besitzen.

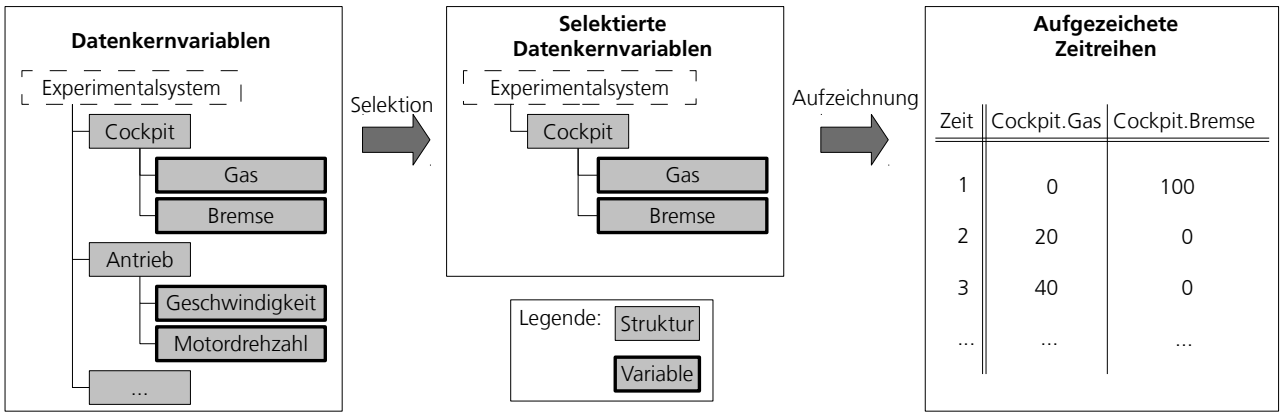


Abbildung 3-9: Selektion und Aufzeichnung von *Dominion*-Datenkernvariablen als Zeitreihe

Für die zur Aufzeichnung markierten Variablen müssen zunächst die Datenkernstrukturen gespeichert werden. Danach werden während des Betriebs die relevanten Variablen als Zeitreihen in einer Tabelle gespeichert (Abb. 3-9). Die Struktur und die Werte bilden die für eine vollständige Reproduzierbarkeit notwendigen Elemente, wobei zur Experimentauswertung nur die Zeitreihen relevant sind. Die Datenkernstruktur wird in Kombination mit den Zeitreihen zur erneuten Bereitstellung aus dem Dominion-DataStore in *Dominion* über ein Abspielmodul benötigt, sodass aufgezeichnete Experimente virtuell erneut durchlebt werden können. In jedem Fall ist die gleiche logische Sicht der Variablen von der Implementierung des Experimentalsystems über den Betrieb und die Aufzeichnung bis hin zur Auswertung gegeben. Somit können verschiedene Personen in unterschiedlichen Rollen im gesamten Lebenszyklus der Daten das gleiche Vokabular bei der Arbeit mit den Variablen verwenden.

Aus dem Dominion-DataStore beziehen zur nachträglichen Auswertung der Experimente Analyseanwendungen (z.B. Ærogator, EAMole, R, DIAdem) die aufgezeichneten Zeitreihen über standardisierte Datenbankschnittstellen (Rechteck mit Strich-Punkt-Linie in Abb. 3-8). Aus Sicht der Analyseanwendungen bildet der Dominion-DataStore die Schnittstelle für den Zugriff auf die aufgezeichneten Daten. Die Anwendungen können Daten aus der Datenbank auslesen und Ergebnisse wieder parallel zu den Rohdaten ablegen. Somit dient der Dominion-DataStore gleichzeitig als Austauschplattform zwischen den Anwendungen für Datenprodukte.

Je nach Integrationsgrad der Anwendung wird auf die semantischen Annotationen über die Integrationsschicht der Architektur (Abb. 3-5) oder über Standard-SQL-Schnittstellen zugegriffen (Abschnitt 4.3). Intensiven Gebrauch von den semantischen Annotationen macht der *Semantic Database Browser* (SDB, Abschnitt 4.2). Mit dem Dominion-DataStore Control Center steht eine Software speziell zur Verwaltung der aufgezeichneten Experimente im Dominion-DataStore bereit, welche zugleich semantische Annotationen zu deren Beschreibung integriert (vgl. Abschnitt 6.3). Beim Auslesen von Daten mittels SQL müssen grundlegende Kenntnisse der Sprache vorhanden sein, dafür bietet SQL jedoch mächtige Möglichkeiten für die gezielte Abfrage der Daten. Als Analyseanwendungen sind weiterhin insbesondere Ærogator und EAMole hervorzuheben, welche im Rahmen von Forschungsaktivitäten entwickelt werden und bei denen eine direkte Integration semantischer Annotationen möglich ist.

3.3 Prozess zur Anreicherung von Daten mit semantischen Annotationen

Nachdem die letzten Abschnitte das Konzept semantischer Annotationen und eine Architektur zur Umsetzung eingeführt haben, wird zum Einsatz der Annotationen ein Prozess für die Anwendung benötigt. Durch Anwendung eines solchen Prozesses können z.B. Daten einer Messfahrt mit eingetretenen Ereignissen in Verbindung gebracht werden, welche die Grundlage für formal modellierte Fahrmanöver bilden. Aufbauend auf die logische Sicht zum Zugriff auf die Messdaten in Abschnitt 3.2.5 wird im Folgenden ein Prozess zum Einsatz semantischer Annotationen auf Messdaten bzw. Massendaten beschrieben. Die in dem genannten Abschnitt beschriebenen Anwendungen und automatisierte Datenanalysen, z.B. in Form von Webservices, arbeiten somit nach der beschriebenen Vorgehensweise beim Einsatz semantischer Annotationen. Sie erlaubt die Beschreibung, Verarbeitung und Auswertung der Daten.

Weiterhin ist dieser Abschnitt parallel zu weiteren (administrativen) Prozessen oder in sie eingebettet zu sehen. Weitgehend unabhängige Prozesse auf Datenbanken bzw. dem Dominion-DataStore beschäftigen sich beispielsweise mit der technischen Administration oder Import bzw. Export von Daten. Als Spezialisierungsfälle des im Folgenden beschriebenen Prozesses zum Einsatz semantischer Annotationen können weiterhin z.B. Data-Mining oder das *Datenqualitätsmanagement* (DQM) betrachtet werden. Bei diesen Anwendungsfällen werden Datenanalysen oder semantische Annotationen aus speziellen Domänen verwendet. Solche Domänen können eigene Prozesse zur Einbindung der Datenverarbeitung und Analyse mitbringen, sodass der folgende Prozess an Stelle der klassischen Datenverarbeitung als Unterprozess integriert werden kann. (vgl. [NBK09a, KN08])

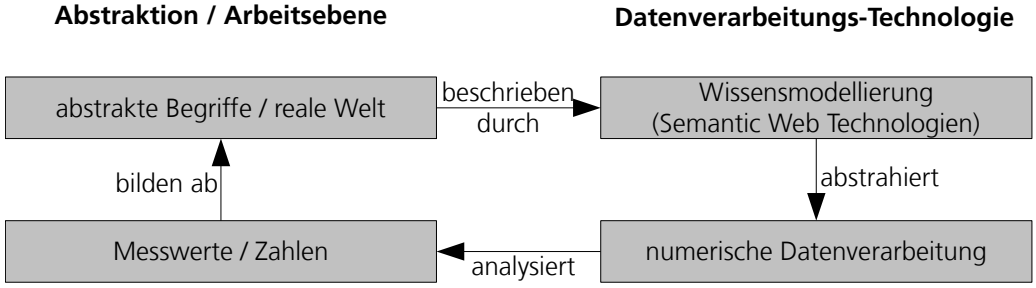


Abbildung 3-10: Abstraktionsebenen für die Datenverarbeitung und Wissensmodellierung

Abbildung 3-10 zeigt die verschiedenen Abstraktionsebenen, auf denen gearbeitet wird, und setzt sie miteinander in Beziehung. Die Ebene der realen Welt und abstrakten Begriffe wird durch Wissensmodellierung beschrieben, wobei Semantic Web Technologien und Ontologien zum Einsatz kommen. In dieser Ebene werden Wissens- und Begriffszusammenhänge erfasst und nach Möglichkeit formal beschrieben. Parallel dazu existiert die Ebene der Messwerte und Zahlen, wobei diese eine Repräsentation der realen Welt darstellen sollen und dieses Abbild z.B. durch den Einsatz von Sensorik erzeugt wird. Messwerte und Zahlen (z.B. in Form von

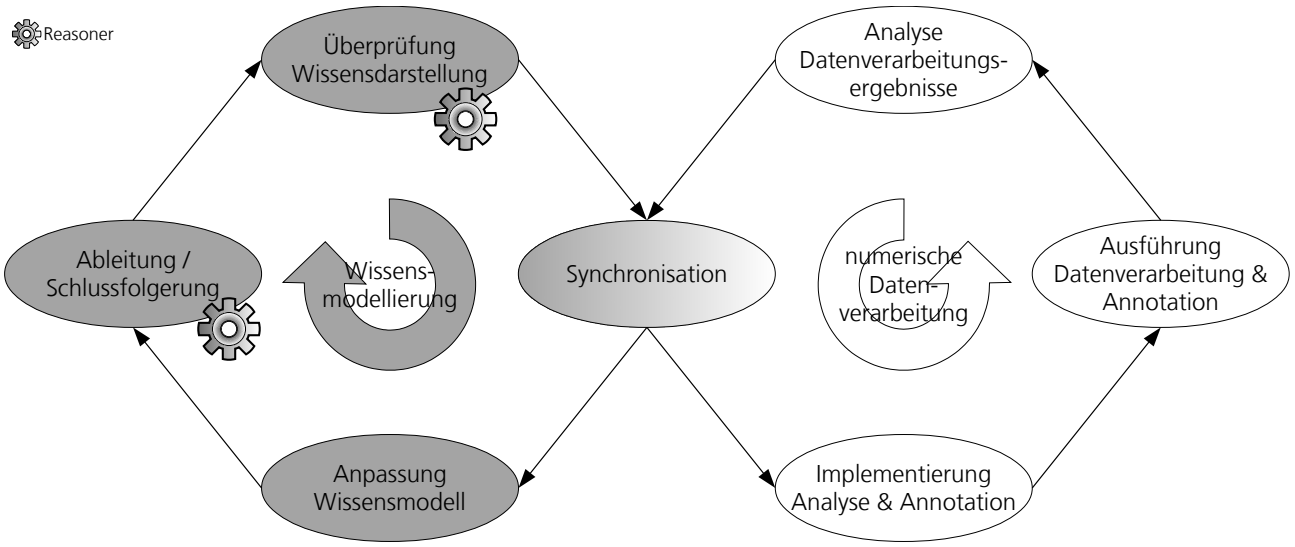


Abbildung 3-11: Prozess zur Anreicherung von Daten mit semantischen Annotationen

Zeitreihen) werden mit Hilfe der klassischen numerischen Datenverarbeitung verarbeitet und analysiert. Es kommen beispielsweise klassische Programmiersprachen, Statistik und Methoden der Regelungstechnik zum Einsatz. Oft ist in dieser Darstellung nicht sofort ersichtlich, was die Zahlen repräsentieren, oder die Darstellung besitzt für einen menschlichen Betrachter nur einen geringen Informationsgehalt (vgl. Abschnitt 3.5). Die Verknüpfung zwischen der numerischen Datenverarbeitung und der abstrahierenden Wissensmodellierung wird realisiert über semantische Annotationen und verbindet diese zwei Ebenen.

An diesem Punkt zur Verknüpfung der zwei Technologie-Ebenen setzt der in Abb. 3-11 dargestellte Prozess zur Anreicherung von Massendaten mit semantischen Annotationen an. Auf der rechten Seite ist die numerische Datenverarbeitung dargestellt. Dort erfolgt die Implementierung klassischer Datenanalysealgorithmen, auf deren Basis relevante Datensätze mit semantischen Annotationen versehen werden. Es folgt die Ausführung der implementierten Algorithmen, welche manuell angestoßen werden kann oder automatisiert durch den Einsatz von SWS (Abschnitt 2.2.3). Als Ergebnis der Ausführung können sowohl semantische Annotationen als auch Zahlenreihen entstehen, welche bei weiteren Schritten der numerischen Datenverarbeitung prozessiert werden können. Während der Analyse werden die berechneten Ergebnisse mit den Eingangsdaten zur Überprüfung verglichen, ob die Implementierung korrekt ist.

Während der Synchronisation wird die numerische Ebene mit der Wissensdarstellung abgeglichen und es werden die semantischen Annotationen der Massendaten mit dem Wissensmodell in Bezug gesetzt. Zur Datenexploration können die im folgenden Abschnitt beschriebenen Methoden (Ebenen mit semantischen Annotationen, vgl. Abschnitt 3.4) eingesetzt werden, welche somit einen entscheidenden Beitrag zur Verknüpfung der numerischen Datenverarbeitung und der Wissensmodellierung darstellen. Nach dem Abgleich der beiden Arbeitsebenen erfolgt auf Seite der Wissensmodellierung die Anpassung und Weiterentwicklung des Modells bzw. der Ontologien. Auf dem erweiterten Wissensmodell werden mit Hilfe eines Reasoners Schlussfolgerungen abgeleitet, sodass das implizit vorhandene Wissen dargestellt werden kann. In der

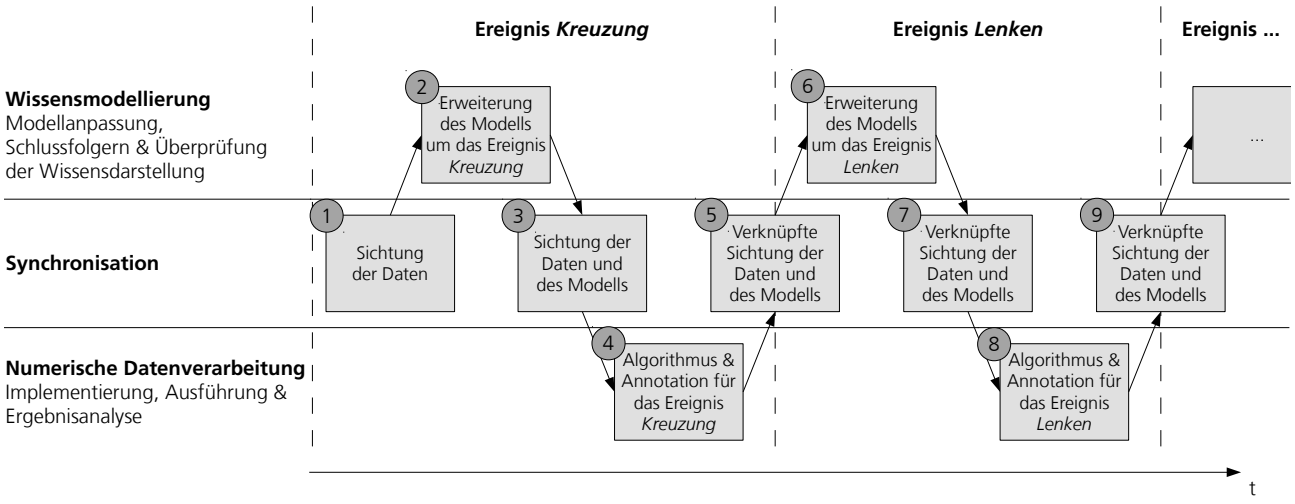


Abbildung 3-12: Semantische Anreicherung von Messdaten mit den Ereignissen $Kreuzung$ und $Lenken$

Abbildung wird der Reasoner-Einsatz durch Zahnräder symbolisiert. Das Modell inklusive der Schlussfolgerungen wird im Anschluss auf Konsistenz und Korrektheit überprüft. Diese Überprüfung erfolgt auf formaler Ebene ebenfalls durch einen Reasoner. Ein menschlicher Anwender überprüft weiterhin, ob die durch den Reasoner durchgeführten Ableitungen den eigenen Erwartungen entsprechen, was insbesondere bei komplexen Darstellungen notwendig ist (vgl. Abschnitt 3.5). Das überprüfte Modell wird als nächster Schritt wieder mit den Massendaten synchronisiert.

Sowohl die Wissensmodellierung als auch die numerische Datenverarbeitung können jeweils beliebig viele Iterationen durchlaufen, bis die entsprechenden Arbeitsebenen den gewünschten Reifegrad besitzen. Die Iterationen müssen nicht zwangsweise abwechselnd erfolgen, sondern können in beliebiger Reihenfolge durchgeführt werden. Die soeben beschriebene Vorgehensweise wird im Folgenden als *Prozess der semantischen Anreicherung (mit semantischen Annotationen)* bezeichnet und wie folgt charakterisiert:

> Der iterative Prozess, bei dem (Massen-)Daten methodisch mit *semantischen Annotationen* verknüpft werden, wird als *semantische Anreicherung* bezeichnet. Bei diesem Prozess realisieren die Annotationen eine Verknüpfung zwischen der numerischen Datenverarbeitung und der Wissensmodellierung.

In Abb. 3-12 ist die beispielhafte semantische Anreicherung von Messdaten aus einem Fahrversuch dargestellt, wobei mit dem Prozess aus Abb. 3-11 die Ereignisse $Kreuzung$ und $Lenken$ identifiziert werden, wenn das Fahrzeug auf eine Kreuzung fährt oder gelenkt wird.

Zunächst erfolgt eine Sichtung der Daten in Schritt 1 und die Einführung des ersten Ereignisses in das Modell (Schritt 2). Da bisher keine Verbindung zwischen den Daten und dem Modell besteht, kann in Schritt 3 jedoch noch keine direkte Verbindung zwischen diesen hergestellt werden. In Schritt 4 wird ein Algorithmus entwickelt (z.B. mit Java), welcher Annotationen für die Messdaten erzeugt, wenn das Ereignis $Kreuzung$ eintritt. Da jetzt neben den Messdaten und dem Modell auch die verknüpfenden Annotationen existieren, können die Daten und das

Modell in den folgenden Synchronisationsschritten 5, 7 und 9 zusammen betrachtet werden, um die Modellebene mit den Messdaten abzugleichen. Die Schritte für das Ereignis $Lenken$ und die folgenden Ereignisse ergänzen wie in den Schritten zuvor das Modell und semantische Annotationen.

Die in diesem Beispiel eingeführten Annotationen für die Ereignisse $Kreuzung$ und $Lenken$ dienen als Grundlage für die weitere beispielhafte Modellierung von Fahrmanövern in Abschnitt 3.5.2 (Abb. 3-17 und 3-18). Um die verknüpfte Betrachtung von Daten und Modell während der Synchronisation zu ermöglichen, werden jedoch zunächst neuartige Methoden benötigt, welche im folgenden Abschnitt eingeführt werden.

3.4 Visualisierung und Interaktion mit semantischen Annotationen

Nach Einführung der behandelten Lösungsansätze zum Einsatz von semantischen Technologien zur Beschreibung von Datenbankinhalten wie z.B. Fahrmanövern, ergibt sich der Bedarf nach Hilfsmitteln für die Visualisierung und Interaktion mit den Ansätzen durch einen menschlichen Anwender.

Dieser Abschnitt stellt in Anlehnung an [NBK11] ein solches Konzept für eine Benutzerschnittstelle zur Visualisierung und Interaktion mit semantischen Annotationen vor. Das Interaktionskonzept soll so allgemein sein, dass es mit verschiedenen annotierten Datenbanken einsetzbar ist. Somit soll es von fachspezifischen Anwendungsfällen unabhängig und als allgemeiner Daten-"Debugger" einsetzbar sein. Daher orientiert es sich grundsätzlich am allgemeinen Prinzip von Datenbank-Browsern, wie es z.B. *Oracle SQL Developer* [Ora11], *pgAdmin* [pgA11] oder *SQuirreL SQL Client* [BWM11] umsetzen, und erweitert dieses um semantische Annotationen. Dieses neuartige Interaktionskonzept wird aus diesem Grund als *semantischer Datenbank-Browser* bezeichnet.

Unabhängig von dem vorgestellten allgemeinen Visualisierungs- und Interaktionskonzept können natürlich semantische Annotationen als Alternative in fachspezifische Anwendungen integriert werden (vgl. Abschnitt 2.2.2). Diese Vorgehensweise ist allgemein üblich für spezialisierte, datenbankbasierte Informationssysteme (vgl.a. [SR13, BG09]). In solch spezialisierten Anwendungen können die technischen Details semantischer Technologien versteckt werden, sodass dem Anwender nicht bewusst ist, auf was für einer technischen Basis er arbeitet. Dagegen wird für die folgenden Überlegungen zur allgemeinen Interaktion davon ausgegangen, dass der Anwender bei Bedarf direkt auf die semantischen Technologien zugreifen kann und will. Nach der Norm EN ISO 9241 [Eur06] zur *Ergonomie der Mensch-System-Interaktion* sind *Aufgabenangemessenheit* und *Erwartungskonformität* zwei wesentliche Faktoren für benutzerfreundliche und effektive Software. Aus diesem Grund werden die folgenden Prinzipien semantische Annotationen möglichst direkt umsetzen und visualisieren.

3.4.1 Ebenen

Eine möglichst direkte Umsetzung ist somit eine Darstellung der Datenbank-Elemente analog zu Datenbank-Browsern bzw. zur Abb. 3-2 mit der gleichzeitigen Kennzeichnung annotierter Elemente. Für die Kennzeichnung muss eine zusätzliche Darstellungsdimension bemüht werden. Die Verwendung der dritten Dimension ist für (zweidimensionale) Tabellen möglich, jedoch mäßig geeignet, da diese auf optischen Hilfseffekten zur Darstellung auf zweidimensionalen Darstellungsflächen (z.B. Monitor) beruht. Außerdem ist die dreidimensionale Darstellung zur Abbildung von *Absolutskalen* ausgelegt. Dagegen ist zur Darstellung von semantischen Annotationen eine *Nominalskala* sinnvoll, um darzustellen, ob überhaupt Annotationen vorhanden sind und ggf. verschiedene Kategorien zu unterscheiden. Die Verwendung von Farben zur Hervorhebung von Tabellen-Elementen ist somit naheliegend. Außerdem kann diese Darstellungsform auf andere Datenbank-Elemente (Datenbank, Schema) für eine einheitliche Visualisierung übernommen werden.

Durch die Verwendung mehrerer Farben können mehrere „Kategorien" unterschieden werden, wobei die verwendbare Anzahl an Farben als begrenzt betrachtet wird, sodass diese noch stets unterscheidbar sind. Die Definition von Kategorien erfolgt interaktiv durch den Benutzer, sodass diese seinen Bedürfnissen für eine konkrete Visualisierung angepasst werden können. Beispielsweise kann der Anwender so für ihn relevante Fahrereignisse oder Fahrmanöver in der Visualisierung hervorheben. Kategorien und deren Darstellung werden durch das Prinzip der *Ebenen* realisiert.

Definition 5 (Ebene semantischer Annotationen) *Eine Ebene L semantischer Annotationen ist ein 6-Tupel (G_o, G_d, f, i, c, b), die durch eine Operation f eine (abgeleitete) Sicht auf einen RDF-Graphen G_o in Form eines neuen Graphen G_d darstellt ($G_d = f(G_o)$). Die Ebene besitzt einen eindeutigen Index i, eine Färbung c und einen textuellen Namen bzw. eine Bezeichnung b. Für f wird eine der folgenden Alternativen angewendet:*

1. *Reasoner*
2. *SPARQL-*DESCRIBE*-Abfrage*
3. *SPARQL-*CONSTRUCT*-Abfrage*
4. *SPARQL-*SELECT*-Abfrage*
5. *Kombination Reasoner mit nachfolgender SPARQL-Abfrage (2 – 4)*

Das Ergebnis einer SELECT-Abfrage ist zunächst eine Menge von URIs, wobei die Auswahl von Elementen mittels SELECT ein häufiger und somit besonders wichtiger Anwendungsfall ist. Die Ergebnismenge wird als Graph lediglich bestehend aus Knoten und ohne Kanten betrachtet. So ist eine einheitliche Behandlung aller Typen abgeleiteter Sichten als Graph möglich.

Für die *Basis-Ebene* gilt $G_o = G_d = G_B$. Sie stellt den Basis-Graphen G_B somit in einer selbständigen Ebene zur Verfügung (vgl. Abschnitt 3.2.3). Als Operation wird ein „Reasoning"

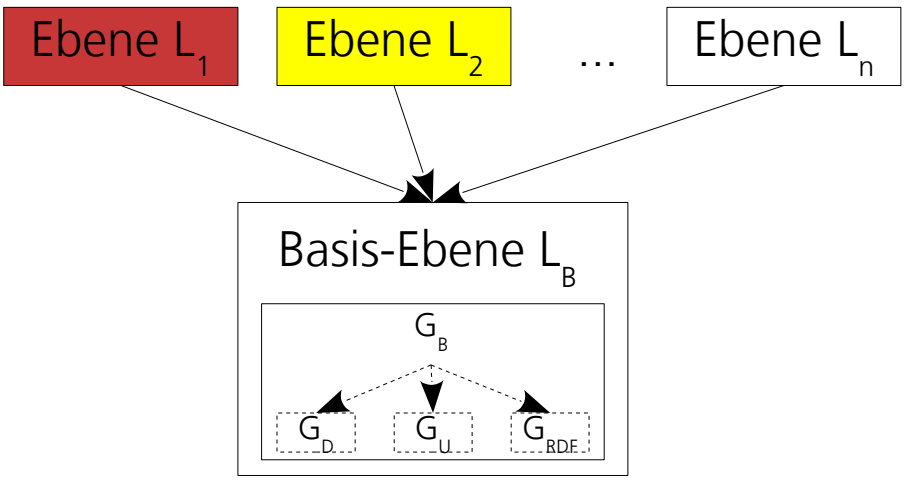


Abbildung 3-13: Abhängigkeit der Ebenen von der Basis-Ebene[2] (frei nach [NBK11])

angewendet, welches die Identität abbildet. Betrachtungsgegenstand für weitere Ebenen ist im Folgenden der Basis-Graph G_B, der die betrachtete Wissensbasis in Form eines RDF- oder OWL-Graphen darstellt und von dem weitere Sichten abgeleitet werden ($G_o = G_B$). Diese Einschränkung ist nicht notwendig, vereinfacht jedoch den Umgang in einer Implementierung und ist für die praktische Anwendung zumeist hinreichend (vgl. Abschnitt 4.2).

Abbildung 3-13 zeigt zwei Ebenen L_1 und L_2, welche in Abhängigkeit von der Basis-Ebene bzw. dem Basis-Graphen definiert wurden. Für eine einfachere Schreibweise gilt die Konvention, dass je Ebenenattribut eine Abbildung existiert, die eine Ebene L auf ihr jeweiliges Attribut abbildet, z.B. $i(L) = i$. Speziell für den Index i gilt weiterhin, dass er tiefgestellt für eine Ebene angegeben werden kann, z.B. L_i mit $i(L_i) = i$. Weiterhin wird mit $\mathbb{L} = \{L_1, \ldots, L_n\}$ die Menge aller n Ebenen bezeichnet.

In Abb. 3-14 werden die zwei Ebenen zur Visualisierung auf eine Tabellendarstellung (Tabellen-Ebene) projiziert. In der linken Hälfte der Abbildung sind die Ebenen über ihren Index i sortiert, sodass die Tabelle mit dem größeren Index weiter oben abgebildet wird. Jedes (farbig dargestellte) Datenbank-Element bzw. dessen Repräsentant als Knoten v im Graphen (z.B. eine Zelle) ist genau dann Element der Ebene L, wenn v ein Element der Knotenmenge V des Graphen G_d der Ebene ist:

$$v \in L :\Leftrightarrow v \in V(G_d(L))$$

Die Färbung c der Ebenen bzw. der enthaltenen Elemente ist in der Abbildung durch Farben und ein Muster der Zellenumrandung (dick / gestrichelt) dargestellt. Bei der Projektion werden die Farben in der finalen Darstellung über die Tabelle gelegt. In der finalen Projektion wird für die Färbung eines Elementes $c(v)$ die Färbung der Ebene $c(L)$ übernommen, wenn die Ebene das Element enthält und den größten Index i dieser Ebenen besitzt:

$$c(v) = c(L_j) :\Leftrightarrow v \in L_j \wedge \forall k\colon v \in L_k \wedge i(L_j) \geq i(L_k)$$

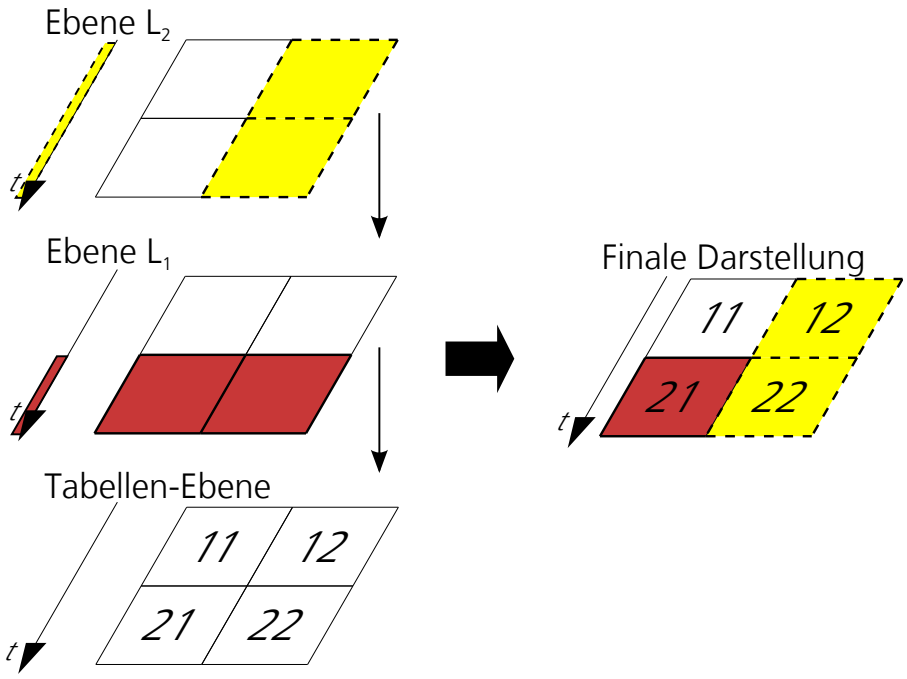


Abbildung 3-14: Projektion von Ebenen auf eine Tabelle[2] (nach [NBK11])

Datenbank-Elemente, die in keiner Ebene vorhanden sind, werden ohne eine spezielle Färbung dargestellt. Durch das *interaktive* Definieren von Ebenen in einem semantischen Datenbank-Browser wird einem Anwender die Möglichkeit gegeben, semantische Datenbank-Annotationen selektiv seinen Bedürfnissen entsprechend zu visualisieren. Mit SPARQL-Abfragen stehen mächtige Ausdrucksmöglichkeiten zur Verfügung. Weiterhin lassen sich durch den Vergleich von Ebenen mit und ohne Reasoning dessen Auswirkungen anschaulich visuell darstellen.

Mittels SELECT-Abfragen erfolgt eine Auswahl von separaten Knoten, welche als Graph interpretiert werden. Es werden somit keine Aussagetripel mehr betrachtet. Außerdem können Kanten als Knoten interpretiert werden, was aber auch bei RDF(S)/OWL möglich ist. Beim Reasoning ist zu beachten, dass einem Graphen ausschließlich neue Kanten hinzugefügt werden. Diese können durch gezielte nachgeschaltete SPARQL-Abfragen dargestellt werden. CONSTRUCT-Abfragen stellen dagegen den einzigen Fall dar, bei dem einem abgeleiteten Graphen optional neue Knoten hinzugefügt werden können.

In der praktischen Implementierung von Ebenen (vgl. Abschnitt 4.2) erhalten diese neben den in diesem Abschnitt eingeführten Attributen ein weiteres in Form eines Aktiv-Flags, über welches Ebenen durch einen Anwender temporär deaktiviert werden können. Ebenen, welche als inaktiv gekennzeichnet sind, werden nicht in der Projektion berücksichtigt.

Das Prinzip von Ebenen zur Überlagerung visueller Informationen wird in einer ähnlichen Art und Weise in *Geographischen Informationssystemen* (GIS) eingesetzt (vgl. [Bun07, S. 31 f.]). Wie im Folgenden dargestellt wird, gibt es jedoch für Ebenen semantischer Datenbank-Annotationen mehrere Möglichkeiten zur Interpretation und Darstellung.

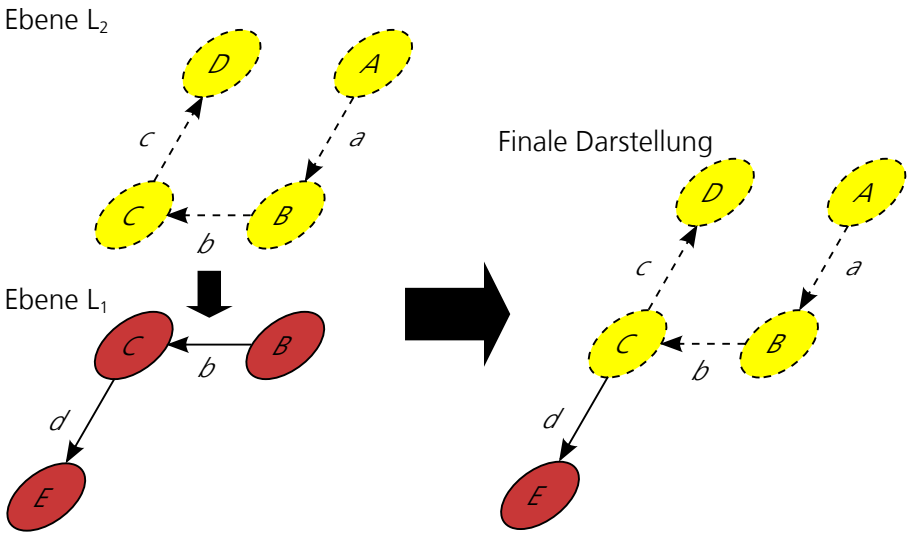


Abbildung 3-15: Überlagerung von Graphen verschiedener Ebenen

3.4.2 Graphdarstellung mit Ebenen

In diesem Abschnitt wird eine alternative Darstellung bzw. Interpretation von Ebenen mit semantischen Annotationen betrachtet. Zuvor wurden Ebenen auf Basis von Datenbank-Elementen visualisiert. Im Folgenden erfolgt diese Visualisierung anhand von Graphdarstellungen. Jede Ebene enthält mit dem abgeleiteten Graphen G_d die zur Darstellung relevanten Informationen.

Abbildung 3-15 zeigt in der linken Hälfte die visualisierten Graphen G_d zweier Ebenen L_1 und L_2 mit einer unterschiedlichen Färbung c. Die enthaltenen Elemente sind daher durch eine Farbe und ein Konturmuster gekennzeichnet. In der rechten Hälfte der Abbildung ist die Vereinigung der beiden Graphen zu sehen, welche durch die Projektion der beiden ursprünglichen Ebenen in die finale Darstellung entsteht. Die Elemente übernehmen die Färbung, welche sich durch das Übereinanderlegen der Elemente ergeben, was nach den im Folgenden beschriebenen Regeln geschieht.

Ein beliebiger Knoten $v \in V$ des Graphen G_d einer Ebene L übernimmt in der finalen Darstellung die Färbung c derjenigen Ebene, in der er enthalten ist und die den größten Index besitzt:

$$c(v) = c(L_j) :\Leftrightarrow v \in L_j \wedge \forall k\colon v \in L_k \wedge i(L_j) \geq i(L_k)$$

Somit handelt es sich bei der Übernahme der Färbung auf einen beliebigen Knoten in der gemeinsamen Projektion mehrerer Graphen um eine Verallgemeinerung der Projektion von Datenbank-Elementen auf eine Tabellendarstellung. Als weiteres Element einer visuellen Graphrepräsentation besitzen die Kanten einer Ebene eine Färbung.

Da bei einem Graphen im vorliegenden Anwendungsfall ein RDF-Modell M nach Def. 3 vorliegt, handelt es sich bei den Kanten um Elemente der Menge der Statements S mit einem Bezeichner aus der Menge der Properties $P \subseteq R$ (vgl. Abschnitt 2.3.1.2). Somit handelt es sich bei der

Betrachtung als Graph um bestimmte Kanten mit einem Bezeichner ($E \subseteq V \times V \times V$, vgl. Def. 2). Eine Kante $e \in E$ ist somit ein Tripel $e = (s, p, o)$, wobei analog zu RDF s Subjekt, p Prädikat und o Objekt genannt werden ($s, p, o \in V$). Die Abbildung $E_1(E) \subseteq V$ liefert die Menge aller Subjekte von E. E_2 und E_3 verhalten sich ebenso für die Prädikate und Objekte.

Analog zu den Knoten übernimmt eine Kante e für die finale Darstellung eine Färbung c derjenigen Ebene L, die den größten Index i besitzt und gleichzeitig in deren Graphen G_d die Kante e enthalten ist. Eine Kante wird genau dann einer Ebene zugeordnet, wenn entweder eine Kante e ein Element von E ist ($e \in E(G_d(L))$) oder aber der Kantenbezeichner $p = p(e)$ als Knoten in G_d verwendet wird. Der Kantenbezeichner p wird genau dann (auch) als Knoten einer Ebene betrachtet, wenn eine der beiden folgenden Bedingungen erfüllt ist. Der Kantenbezeichner p tritt als Subjekt oder Objekt für eine (andere) Kante dieser Ebene auf ($p \in E_1(E)$ oder $p \in E_3(E)$) oder aber wird zumindest in der Knotenmenge V geführt und nicht weiter als Kantenbezeichnung verwendet ($p \in V \wedge p \notin E_2(E)$):

$$\begin{aligned} e \in L \quad :\Leftrightarrow \quad & e \in E(G_d(L)) \\ & \vee\, p(e) \in E_1(E(G_d(L))) \vee p(e) \in E_3(E(G_d(L))) \\ & \vee\, (p(e) \in V(G_d(L)) \wedge p(e) \notin E_2(E(L))) \end{aligned}$$

Somit ergibt sich für eine Kante e deren Färbung $c(e)$ in der finalen Projektion aus folgender Darstellung:

$$c(e) = c(L_j) :\Leftrightarrow e \in L_j \wedge \forall k \colon e \in L_k \wedge i(L_j) \geq i(L_k)$$

Bei der visuellen Darstellung von Graphen ist i.d.R. die praktische Einschränkung zu berücksichtigen, dass zu große Graphen für menschliche Betrachter als unübersichtlich erscheinen. Aus diesem Grund kann in einem solchen Fall die Darstellung auf einen Teilgraphen mit einem maximalen Radius um ein relevantes Element eingeschränkt werden. Für die in diesem Abschnitt beschriebene Färbung und Projektion von Graphen zur Darstellung von RDF/OWL-Inhalten ist weiterhin anzumerken, dass sie vollständig unabhängig von Datenbank-Elementen sind. Die eingeführte Projektion von Graphen kann somit ganz allgemein zur Darstellung RDF/OWL-Inhalten eingesetzt werden.

3.4.3 Belegungsvisualisierung

Eine weitere mögliche Darstellung von Ebenen semantischer Datenbank-Annotationen orientiert sich an der Ordnung betrachteter Daten, wie sie z.B. für Zeitreihen vorliegt, welche über die Zeit sortiert werden. In Abschnitt 3.2.1 wurde bereits gefordert, dass annotierte Tabellen einen eindeutigen Schlüssel bzw. einen Stellvertreterschlüssel in Form der *row_id* besitzen. Über

die *row_id* besitzen die Tabelleninhalte somit die notwendige Ordnung, wobei für Zeitreihen die *row_id* beispielsweise durch Zeitstempel realisiert werden kann.

In Abb. 3-14 ist in der linken Hälfte entlang der Tabellen für die Ebenen eine Zeitachse dargestellt, welche als Abszisse dient und an der die Tabelleninhalte ausgerichtet sind. Basierend auf der Färbung der Tabelleninhalte (Zeilen und Zellen) erfolgt in der Abbildung eine Projektion der annotieren Tabelleninhalte auf die Abszisse. So entsteht jeweils für jede Ebene ein individueller Plot, in welchem Bereiche mit Annotationen gefärbt dargestellt sind. Die so entstehenden *Belegungsvisualisierungen* erinnern entfernt an „Barcodemuster". Für mehrere Ebenen werden die einzelnen Plots übereinander gelegt, sodass die einzelnen Belegungen leicht miteinander verglichen werden können.

Es wird die Projektion für eine Ebene L in die Belegungsdarstellung betrachtet. Bei dieser Projektion werden genau die semantischen Annotationen der Knotenmenge V des Graphen G_d projiziert, welche eine Zeile r_{row_id} oder Zelle $z_{row\_id,column\_name}$ der aktuell betrachteten Tabelle T repräsentieren. Der Index row_id kennzeichnet die referenzierte Zeile und ggf. $column\_name$ die referenzierte Spalte der Tabelle. Dabei entspricht eine Zeile mit dem Index row_id auf der Abszisse a_{row_id} ebenfalls einer Position row_id. Somit wird die Färbung der Achse $c(a_{row\_id})$ für die Zeile row_id genau dann auf die Färbung der Ebene $c(L)$ gesetzt, wenn eine der beiden folgenden Bedingungen erfüllt ist. Entweder liegt r_{row_id} in der Knotenmenge V oder eine Spalte $column\_name$ existiert in der Menge der Spalten C, sodass $z_{row\_id,column\_name}$ in der Knotenmenge V liegt:

$$\begin{aligned} c(a_{row\_id}) = c(L) \quad :\Leftrightarrow \quad & r_{row\_id} \in V(G_d(L)) \\ & \vee \exists column\_name\colon z_{row\_id,column\_name} \in V(G_d(L)) \end{aligned}$$

Letztlich ist anzumerken, dass die in diesem Abschnitt beschriebenen Belegungsplots ähnlich wie die in den Multimediawerkzeugen verwendeten Visualisierungen von Ereignissen aussehen (vgl. Abschnitt 2.2.5, Abb. 2-10). Auch von ihrer Funktion und ihrem Einsatz sind die Belegungsvisualisierungen für semantische Datenbank-Annotationen vergleichbar, jedoch durch das Ebenenkonzept flexibler.

3.5 Anwendung von semantischen Annotationen zur Beschreibung von Zeitreihen

In den letzten Abschnitten wurden verschiedene Aspekte semantischer Technologien zur Beschreibung von Zeitreihen in Datenbanken und deren Visualisierung eingeführt. Für die eingeführten Konzepte besteht nun zur praktischen Anwendung der Bedarf nach entsprechenden Ontologien zur formalen Modellierung auftretender Ereignisse.

Dieser Abschnitt führt in die Verwendung zur Beschreibung von Zeitreihen mittels semantischer Annotationen und Ontologien ein. Somit wird explizit auf Messdaten und ihren zeitlichen Charakter eingegangen, da diese im Vergleich zu statischen Massendaten ohne Zeitcharakter eine Herausforderung darstellen. Zur Umsetzung werden die in Abschnitt 3.3 vorgestellten Prinzipien für den Prozess zur *semantischen Anreicherung* angewendet, um mittels semantischer Annotationen die numerische Datenverarbeitung und die Wissensmodellierung miteinander zu verknüpfen.

3.5.1 Modellierung von Ereignissen in Zeitreihen

Dem Prinzip, Ereignisse in Zeitreihen mittels Ontologien zu beschreiben, liegt die Idee zu Grunde, zunächst mit semantischen Annotationen wesentliche *Elementarereignisse* zu annotieren. Semantische Annotationen werden während einer numerischen Datenverarbeitungsiteration erzeugt. Diese Elementarereignisse bilden die Basis, komplexere Zusammenhänge mittels Ontologien zu modellieren. Auf diese Grundlage bauen weitere Abstraktionsebenen auf, um so immer komplexere Zusammenhänge in Ontologien zu modellieren. So können z.B. Ereignisse aus verschiedenen Bereichen herangezogen werden, um darauf Fahrmanöver zu beschreiben. Auf einer solchen Basis ließen sich beispielsweise Manöversequenzen oder Ressourcenbelegungen nach menschlichen Ressourcen-Modellen (z.B. nach [Wic08]) beschreiben.

Bei einer solchen abstrakten Darstellung können Ontologien ihre Vorteile entfalten, dass sie in Teilontologien aufgeteilt werden können bzw. fremde Ontologien eingebunden werden können. So können bereits vorhandene Modelle leicht wiederverwendet werden oder Experten spezielle Domänen-Ontologien betreuen. Dadurch, dass Ontologien eine formale Beschreibung auf Basis der Prädikatenlogik darstellen, können mathematische Methoden — wie z.B. Reasoning — auf sie angewendet werden.

Beim Einsatz semantischer Annotationen sollten von diesen so viele wie nötig und so wenig wie möglich erzeugt werden. Ein Grund zur Sparsamkeit ergibt sich daraus, dass unnötig viele Modell-Elemente ebenfalls zu erhöhten Laufzeiten beim Reasoning führen. Deshalb müssen bei der Modellierung semantischer Annotationen und darauf aufbauender komplexer Zusammenhänge entsprechende Überlegungen berücksichtigt werden. Ansätze, bei denen Messwerte unmittelbar oder mittelbar in Modell-Elemente umgewandelt werden und somit jede Zeile oder Zelle unmittelbar Annotationen zugewiesen bekommt, scheinen daher für große Datenmengen ungeeignet. Stattdessen ist es erstrebenswert, wirklich nur dort Massendaten zu annotieren, wo relevante Ereignisse auftreten. Eine solche Modellierung besitzt gleichsam einen höheren Abstraktionsgrad, wie es für eine Wissensmodellierung angebracht ist. Idealerweise nähert sich das tatsächliche Verhältnis von Annotationen zur Anzahl annotierbarer Datenbank-Elemente (Zeilen, Zellen) einer Zeitreihe der durch diese Arbeit definierten *semantischen Dichte* an:

> Die *semantische Dichte* ist das Verhältnis der Anzahl möglicher *relevanter* Aussagen (bzw. semantischer Annotationen) zu einem Satz Massendaten in Bezug zur Anzahl vorhandener Elemente (Zeilen, Zellen), über welche potentiell Aussagen getroffen werden können.

Die semantische Dichte gibt somit das theoretisch optimale Verhältnis von semantischen Annotationen zu einem Satz Massendaten an. Die Relevanz von Aussagen ist jedoch immer vom Betrachtungsgegenstand abhängig. Somit trifft die semantische Dichte für logische bzw. semantische Inhalte eine ähnliche Aussage über den Informationsgehalt, wie die Entropie nach Shannon [Sha48] für Zeichenfolgen.

Wird beispielsweise eine Stunde lang die Sensorik eines vor einer Wand stehenden Fahrzeuges aufgezeichnet, fällt eine umfangreiche Zeitreihe an, in der aufgrund von Sensorrauschen sogar die Messwerte variieren würden. Dennoch würden aus Sicht zur Beschreibung von Fahrmanövern keine relevanten Ereignisse auftreten, sodass die semantische Dichte Null wäre.

Ein geeignetes Mittel zur ausdrucksstarken Beschreibung von Zusammenhängen von Elementarereignissen in Zeitreihen sind die bereits geschilderten Zeitbeziehungen nach [All83] (vgl. Abschnitt 2.3.7.1). Zur Umsetzung werden Ereignisse auf das Konzept von *Zeitpunkten* mit einem numerischen Zeitattribut abgebildet. Zwei Zeitpunkte können als Start- und Endpunkt von *Intervallen* herangezogen werden. Zwischen Intervallen können dann 13 Relationen abgefragt werden (Abschnitt D.13). Zu diesen Konzepten gehören beispielsweise, ob Zeitpunkte in einem Intervall liegen oder sich Intervalle auf unterschiedliche Weise berühren oder überlappen. Mittels OWL und SWRL lassen sich diese Beziehungen direkt nachbilden, sodass sie über ein Reasoning berechnet werden können (vgl. Abschnitt 2.3.4). SWRL ist für diesen Anwendungsfall von essentieller Bedeutung, da es sogenannte *Built-Ins* umfasst. Diese Built-Ins erlauben es, Vergleiche von Zahlen in OWL-Ontologien zu berücksichtigen, wie es für den Vergleich der Zeitpunkte über ein Zeitattribut notwendig ist.

Abbildung 3-16 zeigt basierend auf der Architekturdarstellung in Abb. 3-5, wie die Verarbeitung der Ontologie mit den Ereignissen berücksichtigt wird. Das Reasoning auf der in den Hauptspeicher geladenen Ontologie wird durch einen Reasoner durchgeführt, welcher über das Semantic Web Framework eingebunden ist.

3.5.2 Einsatz von Automaten zur Beschreibung von Zeitreihen

Endliche Automaten sind ein geeignetes Mittel, um Zustandsänderungen als Reaktion von Eingabe- bzw. Ereignissequenzen formal und generisch zu beschreiben. Automaten bilden ein leicht verständliches aber dennoch leistungsfähiges Werkzeug, da sie ein Gedächtnis besitzen und somit eine sinnvolle Ergänzung zu reiner Logik darstellen. Weiterhin können sie visuell anschaulich dargestellt werden, was deren Verständnis und Anwendung unterstützt. Aus diesem

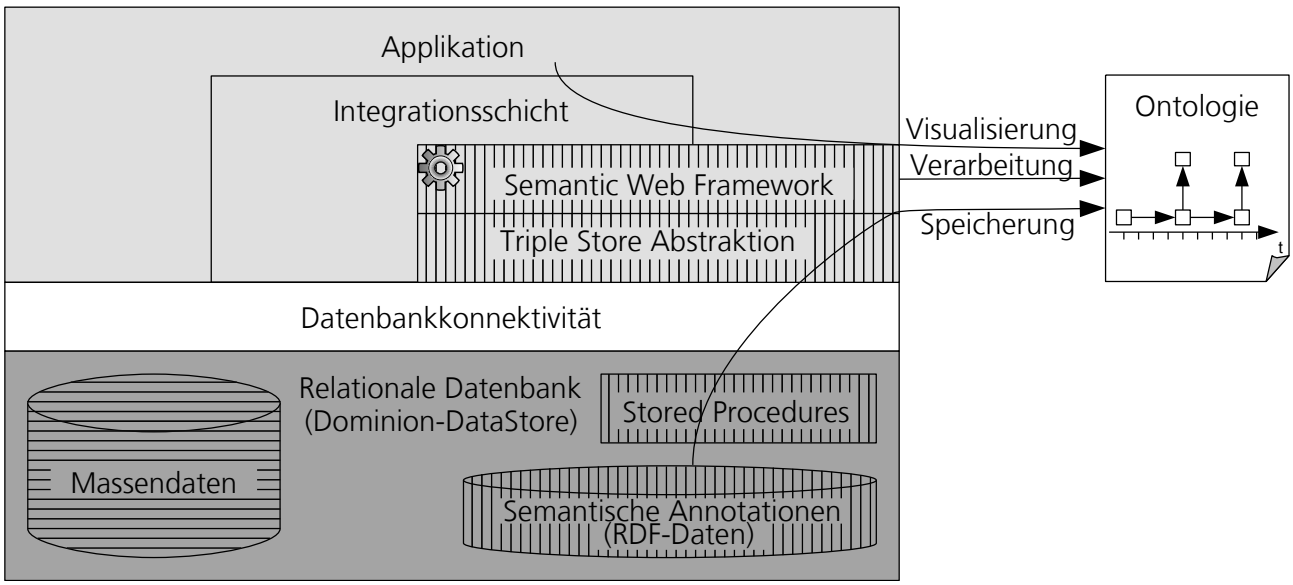


Abbildung 3-16: Ontologie mit Ereignissen in der Architektur

Grund werden Automaten in vielfältigen Einsatzbereichen verwendet und die im Folgenden diskutierten Methoden können leicht für andere Anwendungsfälle wiederverwendet werden.

Automaten und deren Eigenschaften werden in der theoretischen Informatik ausführlich betrachtet, weswegen ihre Eigenschaften gut verstanden werden. Beispielsweise können Petri-Netze auf Automaten abgebildet werden und umgekehrt (vgl. [Lun03, Kap. 11.5, S. 363]). Die Möglichkeit von Automaten, Verhalten generisch zu beschreiben, hat dazu geführt, dass sie ebenfalls in Form von *Zustandsdiagrammen* in der *Unified Modeling Language* (UML) Einzug gehalten haben (vgl. [Obj12b, Kap. 15, S. 541 ff.]). In dieser Form werden Automaten z.B. zur Modellierung von reaktiven Systemen und eingebetteten Systemen eingesetzt, z.B. in [Noy11]. Im Folgenden werden Automaten eingeführt, um mit ihrer Hilfe semantische Annotationen bzw. Elementarereignisse von Zeitreihen zu verarbeiten. Auf dieser Basis können im weiteren Verlauf Fahrmanöver modelliert werden.

Auch in [ZH08] werden Automaten zur Verarbeitung von Ereignissequenzen mittels OWL eingesetzt. Jedoch wird dort nach jeder Änderung der Umwelt das Modell *von außen* angepasst, um einen Automaten zu realisieren und anschließend auf dem angepassten Modell ein Reasoning durchzuführen. Somit werden immer nur Momentbetrachtungen formalisiert dargestellt und es werden keine zeitlichen Ausdehnungen betrachtet, wie sie für die umfassende Betrachtung von Zeitreihen relevant sind. Ebenfalls greift die auf OWL aufbauende Spracherweiterung *tOWL* [FMK10] zur Beschreibung von zeitlichen Änderungen auf Automaten zurück (vgl. Abschnitt 2.3.7.1). Zur Abbildung der Features von Automaten in Ontologien, werden dort jedoch explizite Spracherweiterungen für OWL eingeführt, sodass mit einem handelsüblichen Reasoner (z.B. Pellet) solches Wissen nicht verarbeitet werden kann. Trotz Literaturrecherche ist bei Entstehung dieser Arbeit keine weitere Arbeit bekannt, welche Automaten wie im Folgenden allein mit OWL/SWRL umsetzt, sodass Zeitreihen über einen Zeitraum hinweg beschrieben werden und dass ein handelsüblicher Reasoner den Automaten berechnet.

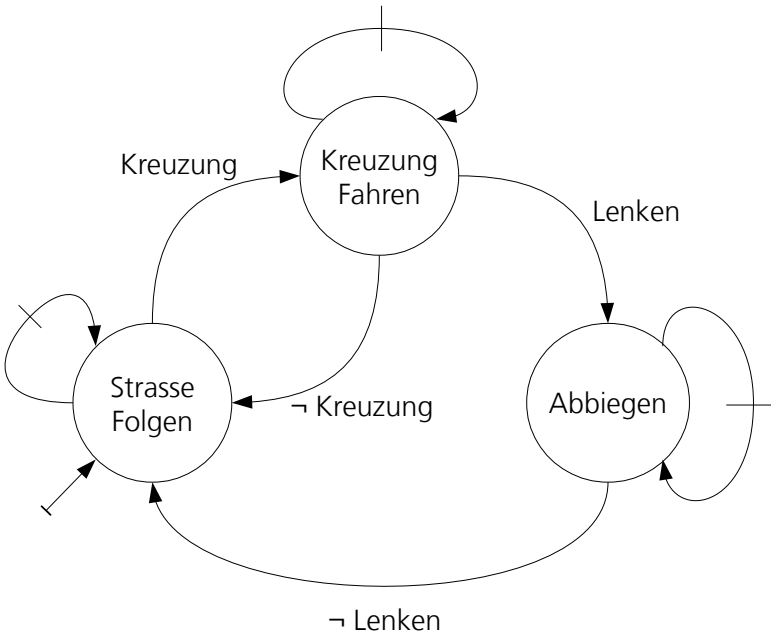


Abbildung 3-17: Beispielautomat zur Beschreibung des Abbiegens auf Kreuzungen

Da für die betrachteten Probleme die Akzeptanz von Eingaben nicht relevant ist, ergibt sich statt einem sonst üblichen 5-Tupel [CLR01, Kap. 34.3, S. 862 ff.] eine reduzierte Beschreibung. Ein deterministischer endlicher Automat A ist somit im Sinne dieser Arbeit ein 4-Tupel (Q, Σ, δ, q_0), wobei gilt:

- Q ist eine endliche Menge von *Zuständen*
- Σ ist ein endliches *Eingabealphabet*
- δ ist eine Funktion von $Q \times \Sigma$ nach Q und heißt *Transitionsfunktion*
- $q_0 \in Q$ ist der *Startzustand*

Abbildung 3-17 zeigt einen einfachen Automaten, der das Fahren über Kreuzungen und das dortige Abbiegen beschreibt. Das Beispiel basiert auf der Annotation von Ereignissen in dem Prozess zur semantischen Anreicherung aus Abb. 3-12. Die Kreise in Abb. 3-17 repräsentieren die Zustände des Automaten und die Transitionsfunktion ist über Pfeile mit den entsprechenden Eingaben dargestellt. Die formale Beschreibung des Automaten befindet sich in Abschnitt D.14.

In Abb. 3-18 wird der Automat aus Abb. 3-17 in Form einer OWL/SWRL-Repräsentation für eine Zeitreihe angewendet. Die Zeitachse unten im Bild symbolisiert eine in einer Tabelle aufgezeichneten Zeitreihe, wobei die Zeilen einzelnen Zeitpunkten auf der Achse entsprechen. Während einer numerischen Datenverarbeitung der Massendaten werden die Elementarereignisse (in Form von Eingaben) detektiert, sodass semantische Annotationen für ausgewählte Zeilen erzeugt werden.

Für eine bessere Übersichtlichkeit werden statt vollständiger URIs für die Zeilen (vgl. Abschnitt 3.2.1) verkürzte Bezeichner verwendet. Beispielsweise könnten die Annotationen für das Lenken erzeugt werden, wenn bestimmte Grenzwerte für den Lenkradwinkel überschritten

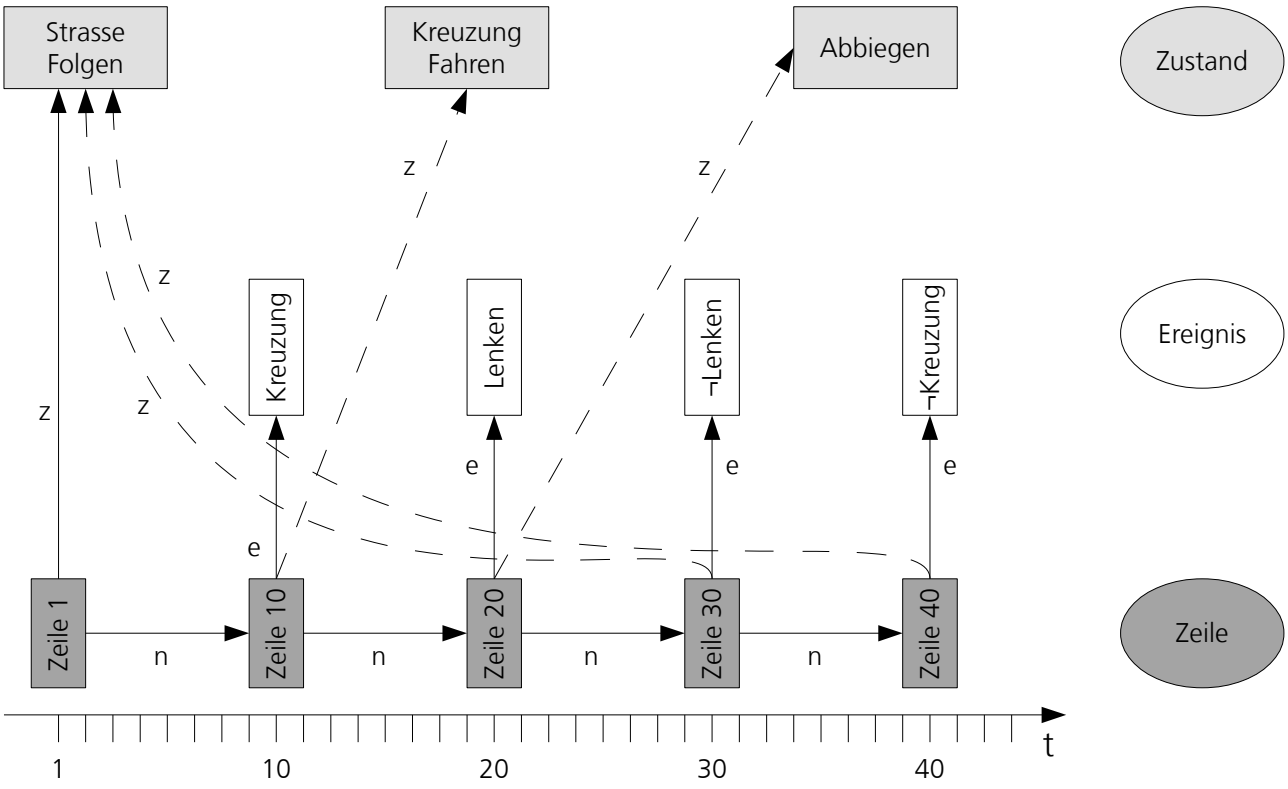


Abbildung 3-18: Anwendung des in OWL/SWRL umgewandelten Automaten aus Abb. 3-17 auf eine Zeitreihe[2] (frei nach [NBK11])

bzw. unterschritten werden. Auf diese Weise entstehen die Instanzen vom Typ `Zeile` für relevante Zeitpunkte. Statt der einfachen Verwendung von Annotationen auf Zeilenbasis könnte eine komplexere Modellierung mit der zusätzlichen Referenzierung einzelner Zellen in einem konkreten Anwendungsfall von Interesse sein, ist jedoch für das folgende Reasoning nicht relevant.

Die `rdf:type`-Attribute für die Zeilen-Instanzen auf das Zeilen-Konzept sind der Übersichtlichkeit wegen nicht dargestellt. Stattdessen sind in Abb. 3-18 die Instanzen jeweils auf einer horizontalen Ebene dargestellt, welcher rechts im Bild ein Konzept zugeordnet wird. Die `Ereignis`- und `Zustand`-Konzepte entsprechen den Eingaben und Zuständen des ursprünglichen Automaten. In einer realen Umsetzung würde weiterhin auf Schema-Ebene über Range- und Domain-Beziehungen die korrekte Verwendung der einsetzbaren Attribute formuliert, sodass eine inkonsistente Nutzung durch den Reasoner entdeckt wird.

Die Attribute `n`, `e` und `z` in der Abbildung stehen als Abkürzung für `naechsteZeile`, `ereignis` und `zustand`. Beim Erstellen der Annotationen werden die `e`- und `n`-Attribute erzeugt, weswegen die Pfeile durchgezogen dargestellt sind. So werden die Ereignisse und deren Sequenz modelliert. Weiterhin wird ausschließlich für die erste Zeile der Zustand annotiert, was der Festlegung des Startzustandes q_0 entspricht.

Die gestrichelten `z`-Annotationen können durch Reasoning berechnet werden, was der automatischen Ausführung des Automaten entspricht. Hierfür ist es notwendig, die Transitionsfunktion δ des Automaten in der Ontologie als SWRL-Ausdrücke zu hinterlegen. Die Darstellung

```
<Transition1>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                   swrl:argument1 <x1> ;
                   swrl:argument2 <Strasse_folgen> ;
                   swrl:propertyPredicate
                         <z>
                 ] [ a       swrl:IndividualPropertyAtom ;
                   swrl:argument1 <x1> ;
                   swrl:argument2 <x2> ;
                   swrl:propertyPredicate
                         <n>
                 ] [ a       swrl:IndividualPropertyAtom ;
                   swrl:argument1 <x2> ;
                   swrl:argument2 <Kreuzung> ;
                   swrl:propertyPredicate
                         <e>
                 ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                   swrl:argument1 <x2> ;
                   swrl:argument2 <Kreuzung_fahren> ;
                   swrl:propertyPredicate
                         <z>
                 ]) .
```

Listing 3-2: Beispieltransition des Automaten zu Abb. 3-17 in Turtle-Notation

erfolgt in *Human Readable Syntax* [HPSB+04, Kap. 2.2]:

$$\mathrm{z}(?x_1, \mathit{Strasse\_Folgen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \mathit{Kreuzung}) \Rightarrow \mathrm{z}(?x_2, \mathit{Kreuzung\_Fahren})$$
$$\mathrm{z}(?x_1, \mathit{Kreuzung\_Fahren}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \neg\mathit{Kreuzung}) \Rightarrow \mathrm{z}(?x_2, \mathit{Strasse\_Folgen})$$
$$\mathrm{z}(?x_1, \mathit{Kreuzung\_Fahren}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \mathit{Lenken}) \Rightarrow \mathrm{z}(?x_2, \mathit{Abbiegen})$$
$$\mathrm{z}(?x_1, \mathit{Abbiegen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \neg\mathit{Lenken}) \Rightarrow \mathrm{z}(?x_2, \mathit{Strasse\_Folgen})$$

Die weiteren Ausdrücke bilden jeweils die Standardtransitionen ab:

$$\mathrm{z}(?x_1, \mathit{Strasse\_Folgen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \mathit{Lenken}) \Rightarrow \mathrm{z}(?x_2, \mathit{Strasse\_Folgen})$$
$$\mathrm{z}(?x_1, \mathit{Strasse\_Folgen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \neg\mathit{Lenken}) \Rightarrow \mathrm{z}(?x_2, \mathit{Strasse\_Folgen})$$
$$\mathrm{z}(?x_1, \mathit{Strasse\_Folgen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \neg\mathit{Kreuzung}) \Rightarrow \mathrm{z}(?x_2, \mathit{Strasse\_Folgen})$$
$$\mathrm{z}(?x_1, \mathit{Kreuzung\_Fahren}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \mathit{Kreuzung}) \Rightarrow \mathrm{z}(?x_2, \mathit{Kreuzung\_Fahren})$$
$$\mathrm{z}(?x_1, \mathit{Kreuzung\_Fahren}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \neg\mathit{Lenken}) \Rightarrow \mathrm{z}(?x_2, \mathit{Kreuzung\_Fahren})$$
$$\mathrm{z}(?x_1, \mathit{Abbiegen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \mathit{Lenken}) \Rightarrow \mathrm{z}(?x_2, \mathit{Abbiegen})$$
$$\mathrm{z}(?x_1, \mathit{Abbiegen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \mathit{Kreuzung}) \Rightarrow \mathrm{z}(?x_2, \mathit{Abbiegen})$$
$$\mathrm{z}(?x_1, \mathit{Abbiegen}) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \neg\mathit{Kreuzung}) \Rightarrow \mathrm{z}(?x_2, \mathit{Abbiegen})$$

In den aufgeführten Ausdrücken erfüllen die $?x_{1,2}$ die Aufgabe von Variablen. Der prädikatenlogische Charakter der Ausdrücke ist deutlich erkennbar. Listing 3-2 zeigt die erste der soeben aufgeführten Transitionen als Teil eines RDF-Modells in Turtle-Notation. Der vollständige Automat ist im Anhang (Abschnitt D.14) zu finden. In Abb. 3-18 ist weiterhin erkennbar, dass nach dem Reasoning die Annotationen von Zeilen zu Zuständen die Definition semantischer Anno-

tationen erfüllen und somit in weiteren Schritten für komplexere Beschreibungen als Eingaben verwendet werden können.

Es folgt die allgemeine Transformation eines beliebigen Automaten A in eine OWL/SWRL-Darstellung:

1. Definition einer Klasse `Zustand` und für jeden Zustand $q \in Q$ ein OWL-Individuum vom Typ dieser Klasse
2. Definition einer Klasse `Ereignis` und für jedes Eingabeelement $\sigma \in \Sigma$ ein Individuum vom Typ dieser Klasse
3. Definition einer Klasse `Zeile` bzw. Wiederverwendung eines äquivalenten Konzeptes, für das für eine konkrete Zeitreihe Individuen erzeugt werden (vgl. Abschnitt 3.2.1)
4. Definition der drei *Object-Properties* e, n und z
5. Überführung der Transitionsfunktion δ, wobei für jedes Element $\delta(q, \sigma) = q'$ folgender Ausdruck in OWL formuliert wird:

$$\mathrm{z}(?x_1, q) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{e}(?x_2, \sigma) \Rightarrow \mathrm{z}(?x_2, q')$$

Die zuvor aufgeführten Schritte bilden somit die TBox (Schemawissen). Die Verknüpfung von A mit einer bestimmten Zeitreihe erfolgt durch die Erstellung entsprechender semantischer Annotationen. Diese bilden die ABox (Instanzwissen) und werden nach folgendem Schema erzeugt:

1. Für die erste Zeile r_1 und jede *relevante* Zeile r_{row_id} der Zeitreihe mit einem Ereignis wird ein Individuum erzeugt (vgl. Abschnitt 3.2.1). Zwei aufeinanderfolgende Individuen r_{row_id} und r'_{row_id} werden durch die Eigenschaft n miteinander verbunden: $\mathrm{n}(r_{row\_id}, r'_{row\_id})$
2. Jedes Individuum einer relevanten Zeile r_{row_id} wird mit dem Individuum des dort erkannten Ereignisses σ über die Eigenschaft e verknüpft: $\mathrm{e}(r_{row\_id}, \sigma)$
3. Der ersten Zeile r_1 der Zeitreihe wird der Startzustand q_0 zugeordnet: $\mathrm{z}(r_1, q_0)$

3.5.3 Erweiterung von Automaten zur Beschreibung von Zeitreihen

Aufgrund der Ausdrucksmöglichkeiten von OWL können durchaus komplexere Möglichkeiten zur Beschreibung von Ereignissen in Zeitreihen als einfache Automaten verwendet werden. Dennoch sind Automaten ein geeignetes Konstrukt, um komplexe Ereignisse auf Basis von Abfolgen von Elementarereignissen zu modellieren. Um von den erweiterten Möglichkeiten von OWL Gebrauch zu machen, werden daher im Folgenden einige Konstrukte verwendet, welche OWL und Automaten weiter miteinander kombinieren und für die dieser Arbeit zugrunde liegenden Anwendungsfälle hilfreich sind.

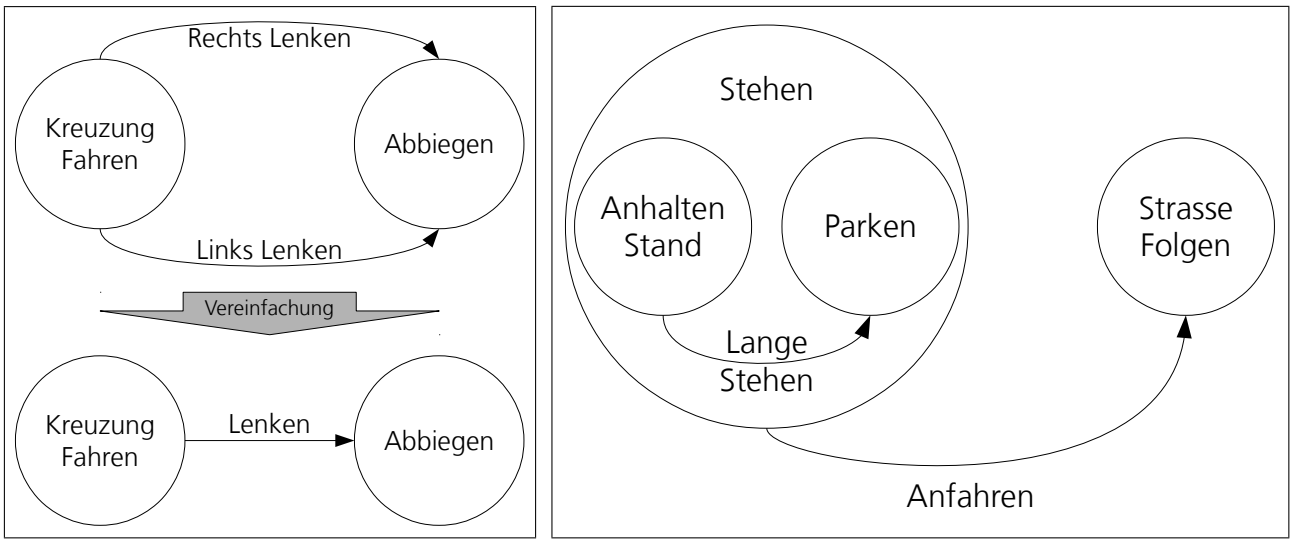
3.5.3.1 Einsatz von Sub-Properties und Unterklassen

Das Beispiel im letzten Abschnitt enthält immer paarweise positive und negative Ereignisse, um zu beschreiben, dass für bestimmte Zeitabschnitte immer entweder die positive oder die negative Eingabe Gültigkeit besitzt. Für diese Ereignisse bietet sich als alternative Modellierung die Einführung eines negierenden Object-Properties an. Somit werden für die Rolle e (Ereignis) zwei Sub-Properties pe (positives Ereignis) und ne (negatives Ereignis) eingeführt. Weiterhin werden die Ereignisindividuen nicht mehr benötigt, welche negative Ereignisse ausdrücken, sodass das obige Beispiel lediglich mit den zwei Individuen Lenken und $\mathit{Kreuzung}$ umgesetzt wird. Zu beachten ist, dass bereits eine inhaltliche bzw. semantische Interpretation des Automaten stattfindet und entsprechend modelliert wird.

In dem Beispiel würden bei dieser Vorgehensweise das Tripel `(Zeile20, e, Lenken)` in das Tripel `(Zeile20, pe, Lenken)` und das Tripel `(Zeile30, e, ¬Lenken)` in die Aussage `(Zeile30, ne, Lenken)` umgewandelt werden. Diese Modellierung wirkt sowohl ausdrucksstärker als auch intuitiver. Als wesentlicher Vorteil ergibt sich aber, dass δ mit weniger Aufwand dargestellt werden kann. So können jetzt zwei Aussagen, welche sich lediglich durch die positive bzw. negative Eingabe (pe bzw. ne) unterscheiden, zu einer Regel zusammengefasst werden, die über die übergeordnete Eingabe e formuliert wird. Beim Reasoning wird dann die Vererbungshierarchie der Rollen aufgelöst und so sowohl bei positivem als auch bei negativem Ereignis die gemeinsame Regel angewendet. Mit dieser Modellierung werden im Beispiel statt 8 Standardtransitionen nur noch 6 benötigt. Je umfangreicher δ ist, bzw. je größer der Anteil der Standardtransitionen ist, desto deutlicher tritt dieser Einsparungseffekt i.d.R. auf.

Eine weitere Möglichkeit zur Verfeinerung der Modellierung ist die Verwendung von Hierarchien für die Eingabeinstanzen. Dann werden zusätzlich zu der Klasse `Ereignis` noch weitere Unterklassen für spezielle Typen von Ereignissen eingeführt. Die konkreten Ereignisindividuen werden dann als Instanzen von den Unterklassen erzeugt. So können bei der Ausformulierung von δ ebenfalls Aussagen zusammengefasst werden, indem statt einzelne Ereignisindividuen direkt ganze Klassen von Individuen in einer Regel abgefragt werden, wie dies beispielsweise Abb. 3-19(a) für das Ereignis Lenken zeigt. Bei der Ausführung übernimmt ebenfalls der Reasoner die Aufgabe zu bestimmen, zu welcher Klasse Individuen gehören und die korrekte Aussage von δ anzuwenden.

Auch für die Zustände lässt sich eine Modellierung mit Unterklassen verwenden. Dies hat zur Folge, dass die gleichen Aussagen von δ für mehrere Zustände gemeinsam verwendet werden können und so ebenfalls der Umfang von δ reduziert werden kann. Abb. 3-19(b) zeigt die gemeinsame Nutzung der Transition $\mathit{Anfahren}$ für die Zustände $\mathit{Anhalten\_Stand}$ und Parken. Abschließend lässt sich festhalten, dass je komplexer ein Automat A ist, desto mehr Sorgfalt muss insgesamt bei der Modellierung von A angewandt werden. So lässt sich eine effiziente Umsetzung mit den in diesem Abschnitt vorgestellten Optimierungen realisieren.



(a) Modellierung von *Links_Lenken* und *Rechts_Lenken* als Sub-Property von *Lenken*

(b) Modellierung der Zustände *Anhalten_Stand* und *Parken* als Unterzustände von *Stehen*

Abbildung 3-19: Möglichkeiten zur Reduktion der Anzahl von Transitionen eines Automaten

3.5.3.2 Kombination mehrerer Eingaben

OWL/SWRL erlaubt komplexe Ausdrücke zur Formulierung der Elemente von δ. Die Darstellung kann dabei so aufwendig werden, dass ein Automat kaum noch zu erkennen ist. Dennoch sind Automaten ein gedanklich so gut erfassbares Hilfskonstrukt, dass weiterhin an ihnen als grundlegende Idee festgehalten wird. In diesem Abschnitt wird die Verarbeitung mehrerer Ereigniseingaben zur Verarbeitung in einem Zustand behandelt. Die Behandlung mehrerer Ereignisse ist besonders praxisrelevant, da bei realistischen Modellen nicht immer davon ausgegangen werden kann, dass Ereignisse nur singulär auftreten und somit mehrere Ereignisse gleichzeitig behandelt werden können müssen.

Für diese Erweiterung wird Σ statt als ein einzelnes Eingabealphabet als Menge von Eingabealphabeten $\{\Sigma_1, \ldots, \Sigma_n\}$ interpretiert. Als Konsequenz wird δ so erweitert, dass sie mehrere Eingaben verarbeitet:

$$\delta : Q \times \Sigma_1 \times \ldots \times \Sigma_n \rightarrow Q$$

Die Überführung der Transitionsfunktion erfolgt für jedes Element $\delta(q, \sigma_1, \ldots, \sigma_n) = q'$ in einen folgenden Ausdruck:

$$\mathrm{z}(?x_1, q) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathfrak{e}_{qq'}(?x_2, \sigma_1, \ldots, \sigma_n) \Rightarrow \mathrm{z}(?x_2, q')$$

Die Konsequenz (consequent bzw. head) auf der rechten Seite des Ausdrucks bleibt unverändert. Die linke Hälfte des Ausdrucks in Form des Bedingungsteils (antecedent bzw. body) besteht aus den durch logische *Und* verbundenen Atomen und dem Term $\mathfrak{e}_{qq'}$. Der Term $\mathfrak{e}_{qq'}$ ist ein Platzhalter, der eine logische Funktion für den Zustandswechsel von q nach q' über mehrere Eingaben beschreibt. Zur Ausgestaltung von $\mathfrak{e}_{qq'}$ können verschiedene OWL/SWRL-Atome

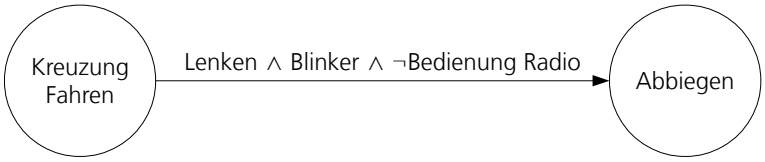


Abbildung 3-20: Beispiel einer Transition mit mehreren Eingaben

beliebig miteinander kombiniert werden. Da in einer Regel die Atome durch *Und* miteinander verknüpft werden, wird ein logisches *Oder* durch zwei Regeln abgebildet, welche sich in dem relevanten Term unterscheiden. Somit können zur Gestaltung der Transitionsbedingung eines Automaten in einer solchen Darstellung beliebige logische Ausdrücke eingesetzt werden, was im Vergleich zu einfachen Automaten eine deutliche Erweiterung darstellt. In Abb. 3-20 ist ein entsprechendes Beispiel einer Transition mit mehreren Eingaben dargestellt.

Zur Beschreibung von $\mathfrak{e}_{qq'}$ und Ableitung der logischen Ausdrücke können bekannte Verfahren aus der Logik wie *Karnaugh-Diagramme* [Vei52, Kar53] eingesetzt werden. Wird so festgestellt, dass für eine Transition nur ein Teil der n Eingaben relevant ist, führt dies dazu, dass $\mathfrak{e}_{qq'}$ entsprechend vereinfacht werden kann. Auf diese Weise kann bei einem solchen Automaten die Transitionsfunktion δ vergleichsweise kompakt dargestellt werden.

Bei dieser Darstellung mit mehreren parallelen Eingaben bzw. Ereignissen mag es sinnvoll sein, eine vorhergegangene Eingabe erneut zu berücksichtigen, wenn sie weiterhin gültig sein soll. Eine Möglichkeit wäre die Berücksichtigung in expliziten Zuständen, was jedoch zu einer Explosion der Zustandsmenge führen kann. Eine andere Alternative ist, über Reasoning vorhergegangene Ereignisse für nachfolgende Zeilen zu übernehmen. Die beiden folgenden Ausdrücke bewirken genau diesen Effekt:

$$\mathrm{pe}(?x_1, ?ereignis) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{keinE}(?x_2, ?ereignis) \Rightarrow \mathrm{pe}(?x_2, ?ereignis)$$
$$\mathrm{ne}(?x_1, ?ereignis) \wedge \mathrm{n}(?x_1, ?x_2) \wedge \mathrm{keinE}(?x_2, ?ereignis) \Rightarrow \mathrm{ne}(?x_2, ?ereignis)$$

Zusätzlich zu den bisher verwendeten Variablen $x_{1,2}$ wird die Variable $ereignis$ verwendet, die das Ereignis bestimmt, welches für die nachfolgende Zeile x_2 von der vorhergehenden x_1 übernommen wird. Aufgrund der OWL zugrunde liegenden *Open World Assumption* kann allein aus der Nicht-Existenz eines Ereignisses für eine Zeile nicht daraus geschlossen werden, dass es nicht existiert. Stattdessen muss dies explizit vermerkt werden, was im vorliegenden Fall durch das Object-Property keinE umgesetzt wird. Allerdings erlauben einige Reasoner die explizite Aktivierung von Closed World Betrachtungen, was in einem solchen Fall sinnvoll sein kann.

3.5.3.3 Bereiche in der Belegungsvisualisierung

Eine weitergehende Betrachtung für Zustände und Ereignisse geht von der Interpretation aus, dass sie zwar punktuell auftreten, aber dennoch für einen gewissen Zeitraum eine Gültigkeit

besitzen (vgl. Abb. 3-18). In Bezug auf Eingaben bietet sich diese Interpretation für die in den letzten Abschnitten beschriebene Darstellung positiver und negativer Eingaben an, dass eine positive oder negative Eingabe solange gültig ist, bis das entsprechende Komplement eintritt.

Aufgrund von dieser Interpretation kann die in Abschnitt 3.4.3 bzw. Abb. 3-14 vorgestellte *Belegungsvisualisierung* entsprechend erweitert werden. Eine solche Darstellung würde nicht nur exakt die Punkte auf der (Zeit-)Achse visualisieren, sondern auch den Bereich dazwischen, wenn er als zusammenhängend identifiziert wird.

Für diese Identifikation bietet sich die explizite Markierung im Wissensgraphen an. In der vorliegenden Arbeit wird hierfür ein separates Konzept `Intervall` eingeführt, welches die Eigenschaften `startPunkt` und `endPunkt` besitzt (vgl. Abb. D-2). Aufgrund dieser `Intervall`-Beschreibung erfolgt später die Visualisierung in der Implementierung (vgl. Abb. 4-5). Bei Bedarf kann eine Regel definiert werden, welche aus den zuletzt betrachteten positiven und negativen Ereignissen diese Intervalle per Reasoning ableitet:

$$\begin{aligned} \mathrm{n}(?x_1, ?x2) \quad &\wedge \quad \mathrm{e}(?x_1, ?ereignis) \\ &\Rightarrow \quad \mathrm{Intervall}(?x_1) \wedge \mathrm{startPunkt}(?x_1, ?x_1) \wedge \mathrm{endPunkt}(?x_1, ?x_2) \end{aligned}$$

3.5.4 Komplexität der Berechnung von Automaten

Um Aussagen über den Zeitaufwand von Problemen treffen zu können, ist es sinnvoll, sich mit deren Komplexität auseinanderzusetzen. Es ist nicht von Interesse, wie lange die Berechnung einer konkreten Implementierung auf einem bestimmten Computer benötigt. Vielmehr ist relevant, wie sich die Laufzeit entwickelt, wenn die Berechnung auf größere Eingabeprobleme angewendet wird. Mit dieser Fragestellung beschäftigt sich die *Laufzeitkomplexität*. Die Komplexität der Laufzeit wird in der Landau-Notation ausgedrückt, welche auch O-Notation genannt wird (vgl. [Knu76, CLR01, Kap. 2, S. 23 ff.]).

Für die Betrachtung der Laufzeitkomplexität von Automaten nach Abschnitt 3.5.2 ist zunächst zu berücksichtigen, welche Parameter Auswirkungen auf deren Berechnung haben können. Für einen Automaten sind diese Parameter die Anzahl der Zustände z und die Anzahl der zu verarbeitenden Ereignisse n. Weiterhin ist die Anzahl der Transitionen t_i relevant, welche von dem Zustand i in einen anderen Zustand überführen. Die Zahl aller Transitionen t beläuft sich somit auf $t = \sum_{m=1}^{z} t_m$.

Zunächst wird die Komplexität für einen einzelnen Zustand betrachtet. Befindet sich der Automat in einem beliebigen Zustand i, müssen für diesen die Transitionen des Zustandes überprüft werden, ob diese erfüllt sind und somit in den entsprechenden folgenden Zustand überführen.

Im schlimmsten Fall müssen sämtliche t_i Transitionen des Zustandes überprüft werden. Weiterhin sei t_{max} die maximale Anzahl an Transitionen, welche ein Zustand besitzt. Somit müssen im allerschlimmsten Fall für einen Zustand t_{max} Transitionsregeln geprüft werden. Für das Prüfen einer Transition fallen weiterhin im schlimmsten Fall die Kosten C_t an.

Je nach konkreter Umsetzung kann es möglich sein, dass zunächst aus den z Transitionsgruppen für die Zustände noch die Gruppe für den aktuellen Zustand gewählt werden muss. Zur Prüfung jedes Zustandes fallen die Kosten C_g an. Nachdem die gesuchte Transitionsregel innerhalb der Transitionsgruppe für den aktuellen Zustand gefunden wurde, welche in den nächsten Zustand überführt, wird der Wechsel des Zustands durchgeführt, welcher mit den konstanten Kosten C_w berücksichtigt wird. Somit ergibt sich die obere Schranke für die Kosten C_z für den Übergang von dem Zustand eines Automaten in den folgenden Zustand als:

$$C_z = z \cdot C_g + t_{max} \cdot C_t + C_w$$

Bei der Ausführung des Automaten werden mehrere Zustände nacheinander abgearbeitet, wobei die zuvor aufgeführten Kosten C_z für jedes verarbeitete Ereignis mit daraus resultierendem Transitionsübergang erneut anfallen. Für n Ereignisse ergibt sich die obere Schranke der Laufzeitkosten C_a für die Ausführung des Automaten somit zu:

$$C_a = n \cdot C_z = n \cdot (z \cdot C_g + t_{max} \cdot C_t + C_w)$$

Für die Betrachtung der Laufzeitkomplexität ist von Bedeutung, in Abhängigkeit von welcher Eingangsgröße die Komplexität betrachtet wird. Im Falle dieser Arbeit wird ein Automat zur Berechnung von Fahrmanövern in einer sehr einfachen (vgl. Abschnitt 3.5.2) und später in einer realistischen Version (vgl. Abschnitt 5.3) vorgestellt. Der Automat ist somit in beiden Fällen unveränderlich und es wird untersucht, wie sich in Abhängigkeit der zu verarbeitenden Ereignisse die Laufzeitkomplexität $C_a(n)$ entwickelt.

Der zuvor aufgestellten Formel ist zu entnehmen, dass n für die obere Schranke mit einer Konstanten multipliziert wird. Im besten anzunehmenden Fall würde die Konstante zwar geringer ausfallen, aber niemals verschwinden. Hieraus ergibt sich in Landau-Notation die Tatsache, dass $C_a(n) \in \Theta(n)$ und somit zur Klasse der *linearen* Laufzeitkomplexität gehört. Die Zugehörigkeit zur Klasse der linearen Laufzeitkomplexität besagt, dass sich für die Verarbeitung von immer mehr Ereignissen die Laufzeit asymptotisch linear entwickelt. Somit kann die Laufzeit von einer linearen Funktion als untere und als obere Schranke eingegrenzt werden. Diese Erkenntnis ist positiv zu bewerten, denn die Verarbeitung größerer Eingabeprobleme ist durch Aufgaben mit linearer Laufzeitkomplexität üblicherweise realistisch zu lösen.

Setzt man die Berechnung von Automaten in Zusammenhang mit dem Reasoning, ergibt sich der Schluss, dass die Berechnung von Automaten durch Reasoning niemals ein besseres Laufzeitverhalten als eine lineare Komplexität aufweisen kann. Tatsächlich muss damit gerechnet

werden, dass ein Reasoning ein weniger günstigeres Verhalten aufweist, da Reasoner durch ihre Allgemeinheit sehr vielfältige Probleme berechnen können, was aber ggf. zum Preis einer nicht optimalen Laufzeit erkauft wird. Diese und ähnliche Fragestellungen werden in speziellen Arbeiten zum Thema Reasoning behandelt, welches eine eigenständige Disziplin darstellt (vgl.a. [WLLB06, BHJV08, UKM+10]).

3.6 Zusammenfassung und Bewertung

3.6.1 Zusammenfassung

In diesem Abschnitt wird zunächst die Vision semantischer Annotationen zur formalen Beschreibung von Massendaten und Zeitreihen in Datenbanken vorgestellt. Nach der Einführung semantischer Annotationen wird deren praktische Umsetzung betrachtet. Auf Basis einer entsprechenden Architektur auf Ebene eines einzelnen Systems als auch als Teil eines verteilten Systemverbundes wird deren Einsatz dargestellt. Weiterhin wird die Einbettung in einen Prozess beschrieben, der die klassische numerische Datenanalyse mit semantischen Annotationen als Prozess der semantischen Anreicherung miteinander vereint. Es folgt eine Interpretation semantischer Annotationen für Datenbank-Elemente, die deren Visualisierung und eine Interaktion mit den Annotationen möglich macht. Über den Vergleich mit deterministischen endlichen Automaten wird gezeigt, wie Zeitreiheninhalte mittels semantischer Annotationen und OWL formal beschrieben werden können.

3.6.2 Bewertung

Die folgende Bewertung der behandelten Inhalte ist unterteilt nach den verschiedenen Aspekten der zugrundeliegenden Forschungsfragen (vgl. Abb. 3-1), wie Messdaten mit Domänenwissen in Form von formalen Modellen verknüpft werden können.

Konzept

Semantische Annotationen stellen ein sinnvolles Bindeglied dar, welches relationale Datenbanken mit Semantic Web Technologien verknüpft. Hierfür werden Datenbank-Elemente dynamisch in URIs abgebildet, sodass sie in RDF-Aussagen eingebunden werden können. Mit ihnen ist es möglich, wesentliche Vorteile aus der Welt der relationalen Datenbanken mit entscheidenden Vorteilen von Semantic Web Technologien miteinander zu kombinieren. Aus diesen Gründen ergänzen semantische Annotationen die numerische Datenverarbeitung methodisch ideal um Aspekte der Wissensmodellierung.

Architektur und Darstellung

Die Architektur berücksichtigt die Skalierbarkeit, indem die Annotationen auf mehrere Teilmodelle zur Speicherung aufgeteilt werden. Aus Sicht eines verteilten Systems können fremde Ontologien sehr leicht referenziert werden oder alternativ werden die Datenbank-Elemente in beliebigen anderen Ontologien referenziert. Durch die leichte Integration von Ontologien unterstützen semantische Annotationen daher eine erstrebenswerte Wiederverwendbarkeit und Kooperation.

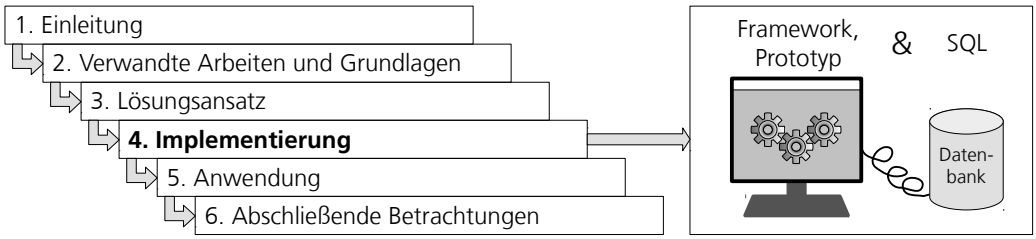
Mit der anschaulichen Visualisierung von Annotationen durch Projektionen wird es möglich, diese leicht nachvollziehbar darzustellen und somit zur interaktiven Exploration einzusetzen. Die beiden zugrundeliegenden Technologien für sich betrachtet besitzen diese Möglichkeit nicht. Damit wird spätestens an dieser Stelle sehr deutlich sichtbar, dass die Kombination dieser beiden Technologien in der vorgestellten Art und Weise viel mehr ist als deren Summe und einen entsprechenden Mehrwert bietet.

Anwendung

Die betrachteten endlichen Automaten zur Beschreibung von Zeitreihen bilden eine generische und sehr effiziente Grundlage, sodass sich auf dieser Basis zeitliches Verhalten beschreiben lässt. Aufgrund ihrer formalen Eigenschaften kann ein Reasoner eingesetzt werden, um über die betrachtete Zeitreihe Aussagen zu treffen. Endliche Automaten können für sich betrachtet keine logischen Aussagen als Eingabe berücksichtigen, sodass der vorgestellte Ansatz leistungsfähiger ist. Da relationale Datenbanken (ohne prozedurale Programmierung) weder Automaten berechnen noch rekursive Berechnungen durchführen können, besitzt der vorgestellte Ansatz aus dieser Sichtweise ebenfalls Vorteile. Weil weiterhin der Umfang von Zeitreihen zu groß und in den meisten Fällen zu redundant ist, ist eine komplette Betrachtung dieser alleine mittels semantischer Technologien ebenfalls nicht sinnvoll. Da diese Eigenschaften mit semantischen Datenbank-Annotationen umgesetzt werden können, bilden die vorgestellten Konzepte eine sinnvolle und sehr leistungsfähige Ergänzung der bereits existierenden Technologien.

Die in diesem Kapitel eingeführten Konzepte sind bisher so generisch gehalten, dass zum jetzigen Zeitpunkt keine Bindung an eine bestimmte Anwendungsdomäne erfolgt. Auch der Bezug zu Zeitreihen ist in vielen Aspekten nicht zwingend notwendig, sodass eine Anwendung in verschiedensten Bereichen möglich ist. Das nächste Kapitel stellt eine Implementierung vor, welche die in diesem Kapitel vorgestellten Konzepte realisiert.

4 Implementierung



In diesem Abschnitt wird eine praktische Umsetzung semantischer Datenbank-Annotationen zur formalen Beschreibung von Zeitreihen vorgestellt, welche im letzten Kapitel als Grundlage eingeführt wurden. Die Anwendung *Semantic Database Browser* wird eingeführt, welche die Konzepte eines *semantischen Datenbank-Browsers* zur Visualisierung und Interaktion mit Datenbank-Annotationen demonstriert. Im Anschluss wird anhand einer konkreten Ontologie gezeigt, wie mit ihrer Hilfe Ereignisse so in einer Zeitreihe beschrieben werden, dass mittels eines Reasoners komplexe Fahrmanöver abgeleitet werden können.

4.1 SIMDa-Framework

Das im Rahmen dieser Arbeit entwickelte SIMDa-Framework (Semantic Integration of Measurement Data) fasst Funktionalitäten zur Arbeit mit semantischen Annotationen für Datenbank-Elemente zusammen, sodass diese leicht wiederverwendet werden können. Die Funktionalitäten orientieren sich an den in Abschnitt 3.2 vorgestellten Architekturkonzepten. Somit realisiert das SIMDa-Framework die *Integrationsschicht* der Architektur in Abb. 3-5 und bildet die Basis für die Umsetzungen in den folgenden Abschnitten dieses Kapitels.

Der Großteil der Funktionalität wird über ein Fassade-Entwurfsmuster [GHJV04, Kap. 4, S. 212 ff.] bereitgestellt, sodass der Anwender mit den meisten Klassen des Frameworks keinen direkten Kontakt hat und die Komplexität in der Benutzung minimiert wird. Für die grundlegenden RDF- und OWL-Funktionalitäten wird auf das Jena-Framework zurückgegriffen und durch das SIMDa-Framework gekapselt (vgl. Abb. 3-5). Folgende Funktionalitäten werden ergänzt:

- Klassen zur Repräsentation von Datenbank-Elementen und zur Konvertierung in URIs bzw. RDF-Ressourcen und zurück werden bereitgestellt (vgl. Abschnitt 3.2.1).

- Klassen zur Kapselung eines *Basis-Graph-Modells* mit der Möglichkeit, Teilmodelle für Datenbanken, Schema und Tabellen zu laden und entladen, werden bereitgestellt (vgl. Abschnitt 3.2.3).
- Es werden abstrakte und konkrete Fabriken [GHJV04, Kap. 3, S. 107 ff.] eingeführt, um den Speicherort für semantische Annotationen zentral steuern zu können. So werden zur Speicherung OWL-Dateien, PostgreSQL und Oracle-Datenbanken mit einer generischen, relationalen Speicherung und Oracle-Datenbanken mit nativer RDF-Speicherung unterstützt. Die Speicherung in Dateien ist zum Testen hilfreich und erlaubt die Behandlung *virtueller*, nicht modifizierbarer Datenbanken auf Basis von CSV-Dateien. Durch die Unterstützung einer generischen, relationalen Speicherung können außerdem beliebige relationale Datenbanken unterstützt werden. Bei Einsatz der nativen Oracle-Speicherung [MAD+05], werden die für Oracle spezifischen Funktionen zur Behandlung von RDF-Modellen eingesetzt. Diese Funktionen erlauben die besonders effiziente Behandlung und Manipulation auf Datenbank-Ebene.
- Klassen zur Repräsentation und Verwaltung von Ebenen semantischer Annotationen werden angeboten. Die Tupel-Definition aus Abschnitt 3.4.1 wird für die praktische Arbeit um Kommentare und ein Aktiv-Flag erweitert. Über das Aktiv-Flag können Ebenen temporär deaktiviert werden. Die Ebenentypen für SPARQL-basierte Ebenen führen ein Caching durch, sodass nicht bei jedem erneuten Zugriff die zugrunde liegenden SPARQL-Abfragen ausgewertet werden müssen. Grundoperationen wie das Suchen von Elementen werden auf der Fassade angestoßen und sämtliche aktive Ebenen werden berücksichtigt.
- Einführung von Proxy-Klassen [GHJV04, Kap. 4, S. 254 ff.] für die RDF-Graph-Elemente (Ressourcen, Properties, Literale) auf die entsprechenden Jena-Klassen. So führt jedes Element eine Referenz auf die Ebene mit, der es zugeordnet ist und kann in einer Darstellung entsprechend eingefärbt werden.
- Mittels entsprechender Abfragen können direkt sämtliche Annotationen für vorgegebene Datenbank-Elemente (z.B. Tabellen) abgerufen werden, sodass sie für eine Visualisierung zur Verfügung stehen.
- Bei der Abfrage von Annotationen für Datenbank-Elemente werden Bereiche bzw. Intervalle nach Abschnitt 3.5.3.3 berücksichtigt, sodass diese direkt geschlossen dargestellt werden können.

4.2 Semantic Database Browser

In diesem Abschnitt wird die Anwendung *Semantic Database Browser* (SDB) vorgestellt, welche die in Abschnitt 3.4 eingeführten Konzepte eines *semantischen Datenbank-Browsers* als Prototyp umsetzt und somit als Machbarkeitsnachweis der in dieser Arbeit vorgestellten Konzepte dient. Die Prinzipien semantischer Annotationen werden direkt umgesetzt, wie es [Eur06] na-

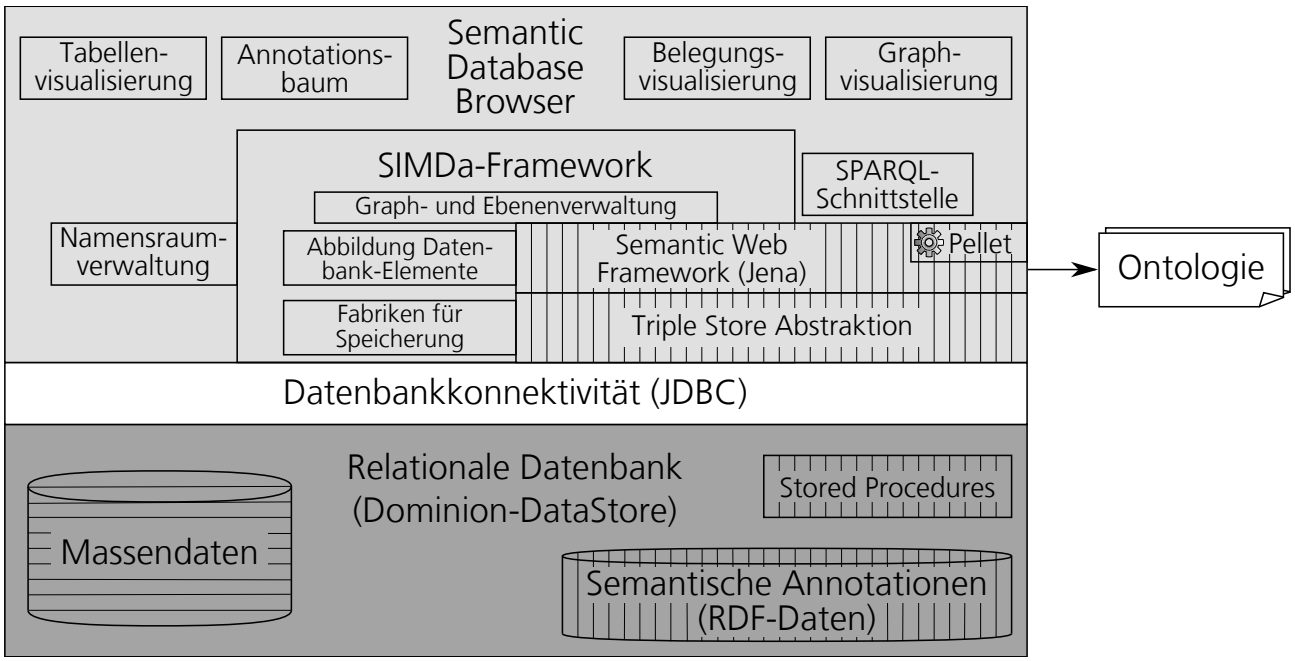


Abbildung 4-1: Architektur des Semantic Database Browsers

helegt. Unabhängig vom Semantic Database Browser ist natürlich weiterhin die Integration der Annotationen in integrierte Informationssysteme möglich.

Die Architektur des Semantic Database Browsers ist in Abb. 4-1 dargestellt und setzt die Referenzarchitektur aus Abb. 3-5 bzw. Abb. 3-16 um. Auf der Applikationsebene werden somit Funktionen zur Präsentation, Visualisierung und Interaktion mit semantischen Annotationen, Datenbank-Elementen und RDF-Graphen bzw. Ontologien bereitgestellt. Für diese Funktionen wird auf die Funktionalitäten darunter liegender Softwareebenen zurückgegriffen.

Abbildung 4-2 zeigt die Anwendung, wobei eine Tabelle zur Darstellung ausgewählt wurde. In dem linken Teilfenster erfolgt die Auswahl der Datenbank-Elemente Datenbank, Schema, Tabelle und Spalte. Annotierte Elemente werden farbig hervorgehoben. Je nach ausgewähltem Datenbank-Element werden die entsprechenden Modelle nachgeladen, sodass immer alle relevanten Annotationen dargestellt werden können und dennoch die Skalierbarkeit der Anwendung erhalten bleibt (vgl. Abschnitt 3.2.3).

Im Hauptfenster wird der Inhalt der ausgewählten Tabelle angezeigt, wobei Spalten, Zeilen und Zellen ebenfalls farbig hervorgehoben werden, wenn sie eine Annotation besitzen. Die Färbung der Elemente erfolgt nach der Ebene, welcher sie zugeordnet werden (vgl. Abschnitt 3.4.1).

In der unteren Hälfte des Bildschirmfotos werden die tatsächlichen Annotationen desjenigen Datenbank-Elementes dargestellt, welches in einem der anderen Fenster gerade ausgewählt wurde (hier die Spalte b). Für die Darstellung wurde eine für RDF-Graphen eher unübliche Baumansicht gewählt. Als Wurzel des Baumes wird die Ressource dargestellt, welche das aktuell ausgewählte Datenbank-Element repräsentiert. Beim Ausklappen des Baumes werden die Tripel sichtbar. In der Baum-Ebene unterhalb von Subjekten befinden sich die Prädikate. In der Ebene unterhalb der Prädikate befinden sich die Objekte. Sowohl Prädikate als auch Objekte

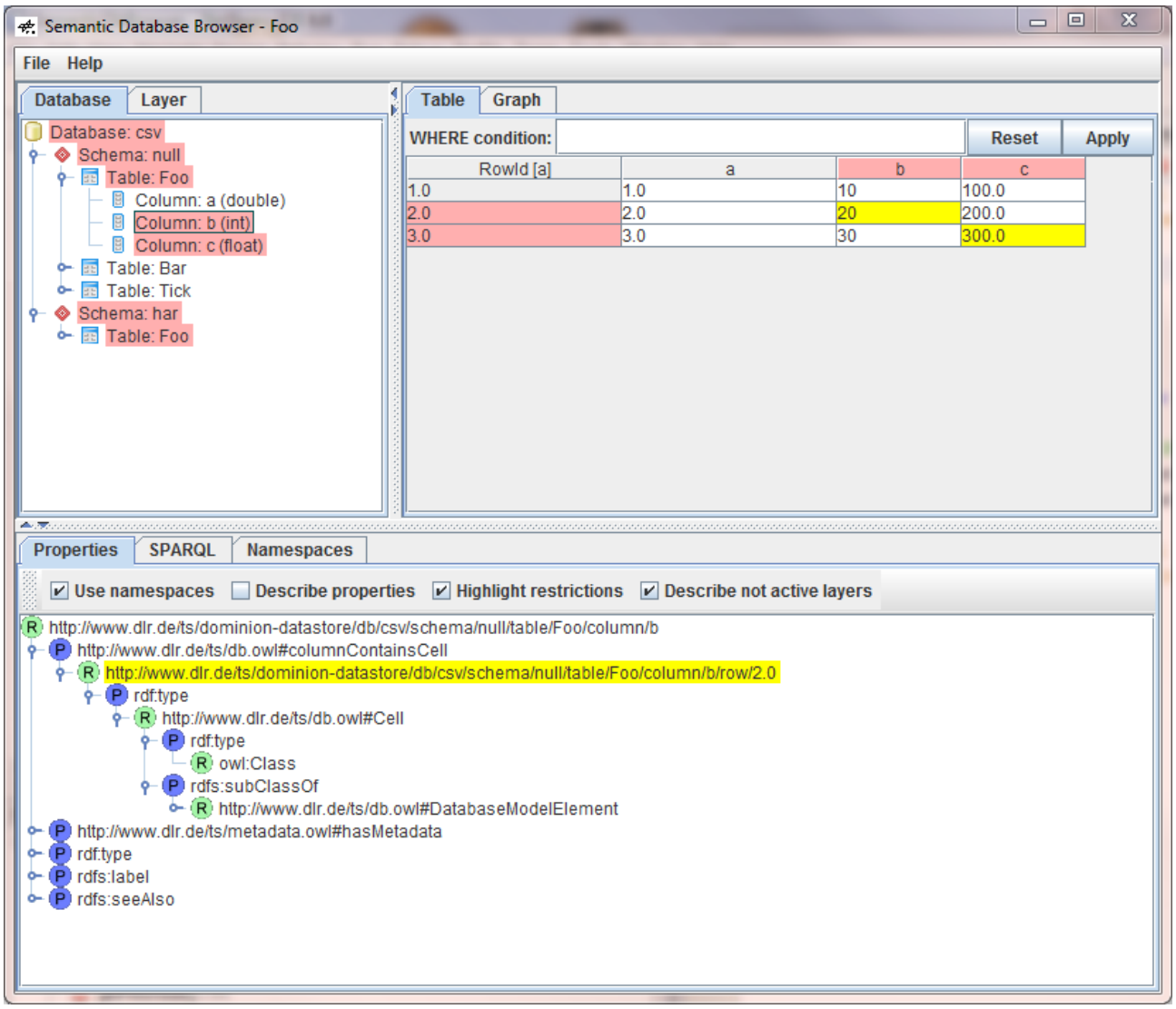


Abbildung 4-2: Semantic Database Browser in der Standardansicht[2] (frei nach [NBK11])

können selbst wieder Subjekte neuer Aussagen sein, sodass die entsprechenden RDF-Elemente ebenfalls auf den darunterliegenden Baum-Ebenen dargestellt werden. Über das Symbol links neben einem Baum-Element wird dargestellt, ob es sich bei dem RDF-Element um eine benannte Ressource (R), Prädikat (P), Literal (L) oder anonyme Ressource (A) handelt.

Gegenüber der häufig gebrauchten Graphdarstellung besitzt die eingesetzte Baumdarstellung einige wesentliche Vorteile. So müssen die Elemente für einen Graphen nicht angeordnet werden (vgl.a. [BETT98]), was Rechenzeit spart. Da nur der wirklich benötigte Ausschnitt des Graphen dargestellt wird, ist diese Darstellung vergleichsweise ressourcenschonend und kann dennoch sehr leicht durch Aufklappen eines Astes erweitert werden. Durch die kompakte Darstellung ist die Baumdarstellung für betrachtete Elemente übersichtlicher und informativer. Jedoch ist zu beachten, dass Zyklen im Graphen beim Aufklappen des Baumes dazu führen, dass bereits besuchte Knoten erneut erscheinen können.

Die Färbung der Baum-Elemente erfolgt ebenfalls nach der Ebenenzugehörigkeit eines Elementes für Graph-Betrachtungen (vgl. Abschnitt 3.4.2). In der derzeitigen Implementierung werden

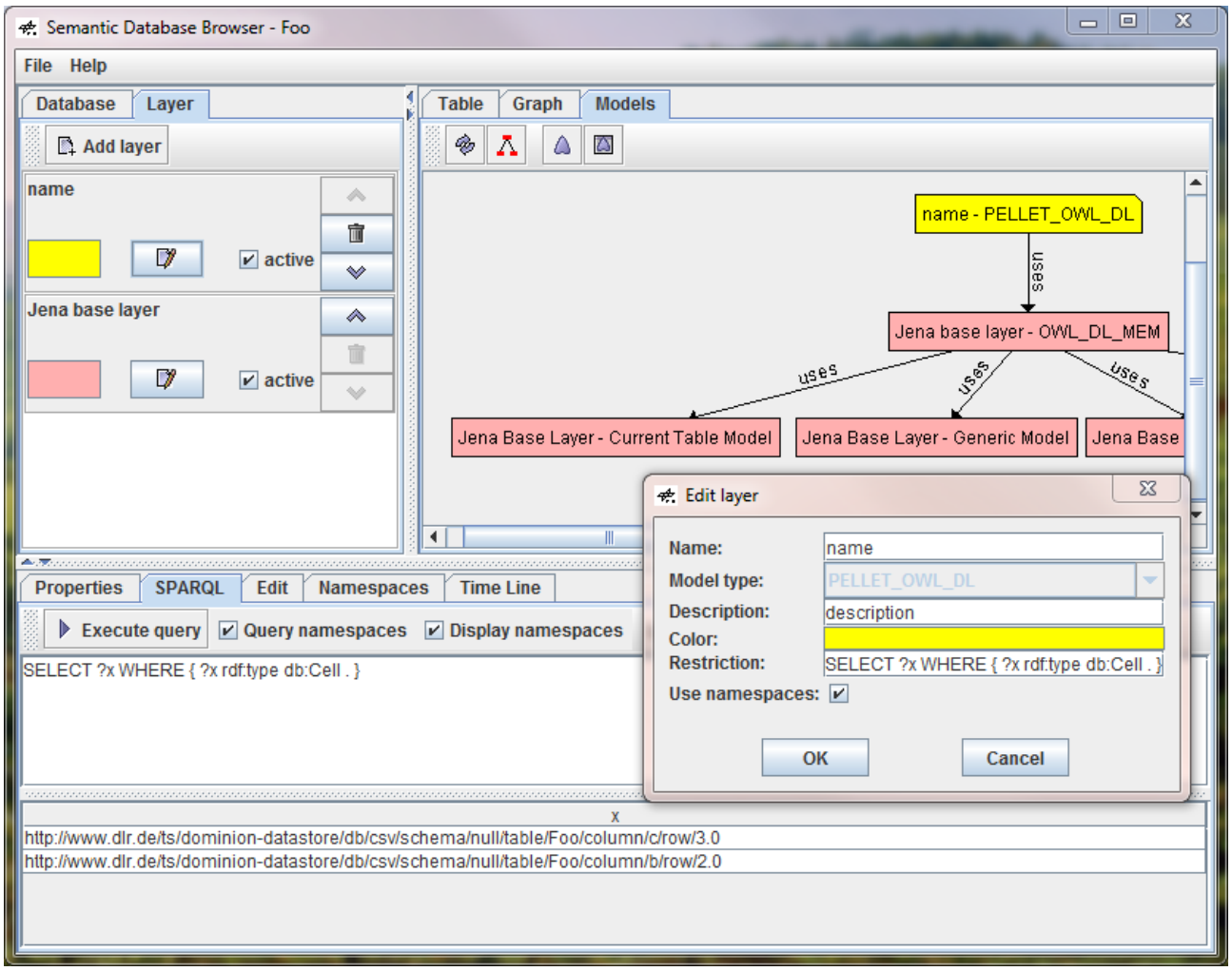


Abbildung 4-3: SDB, Konfiguration von Ebenen

jedoch ungefilterte Ebenen nicht farblich hervorgehoben, da diese den vollständigen Baum einfärben würden und somit kaum ein Informationsgewinn einhergeht.

Als zusätzliches Feature erlaubt die Baumansicht optional die Darstellung von menschenlesbaren Beschriftungen anstatt der URIs, welche in einer Ontologie mittels `rdfs:label` [BG04, Kap. 3.6] als Literal hinterlegt werden. So können Aussagen nahezu als natürlichsprachliche Sätze dargestellt werden. Da Literale eine Sprachangabe besitzen, können Beschriftungen für eine Ressource in mehreren Sprachen in der Ontologie hinterlegt sein. Der Semantic Database Browser stellt jeweils die Beschriftung in der Landessprache dar, welche zu der in der Anwendung ausgewählten am besten passt. Werden in internationalen Teams Ontologien eingesetzt, können diese so für semantische Annotationen durch eine entsprechende Internationalisierung zu einem besseren Verständnis beitragen.

Für die Darstellung einfacher Ressourcen können natürlich registrierte *Namensräume* für eine kompaktere Darstellung optional berücksichtigt werden (Abb. D-3). Aussagetripel können mittels textueller Eingabe zu einem wählbaren Teilmodell hinzugefügt oder entfernt werden, wobei registrierte Namensräume berücksichtigt werden. Weiterhin können RDF-Elemente aus anderen

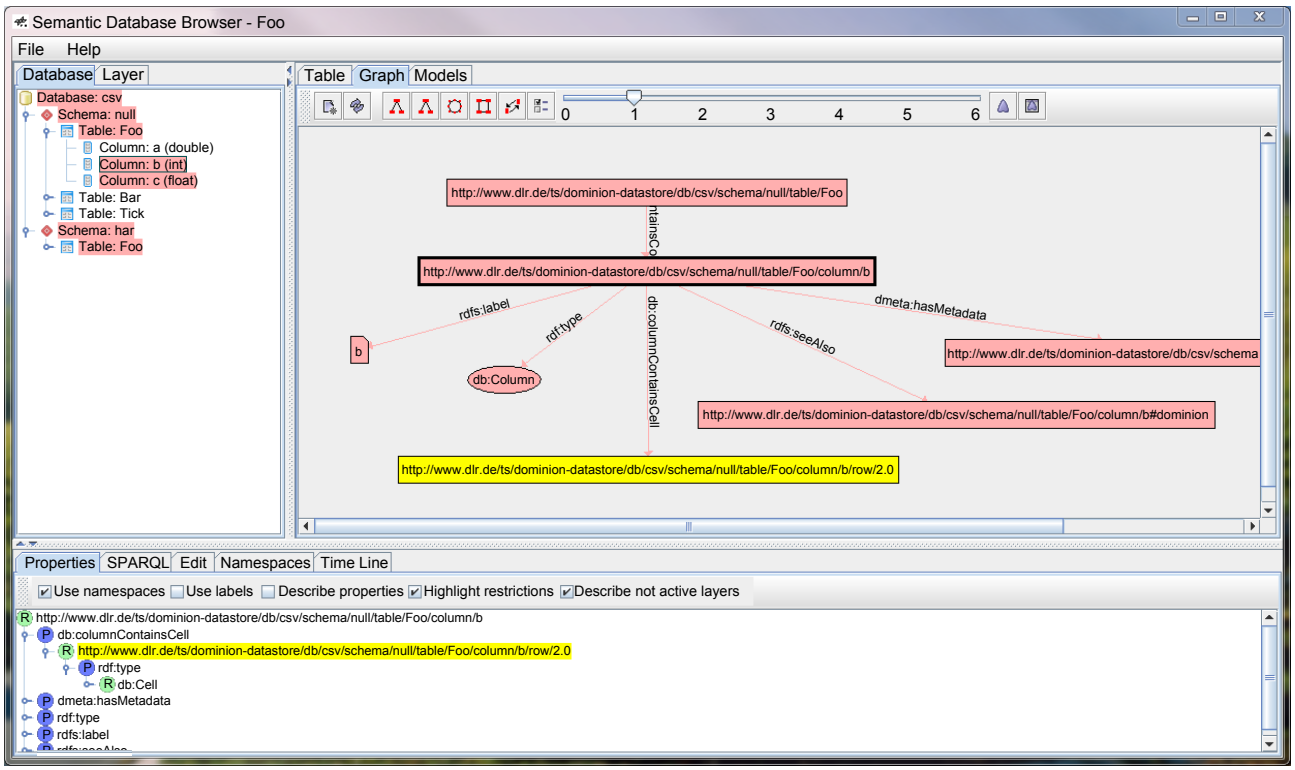


Abbildung 4-4: SDB, Graphdarstellung mit Färbung[4]

Fenstern direkt in das Editierfenster übernommen werden, um in zu editierenden Aussagen berücksichtigt zu werden.

In einem separaten Fenster werden SPARQL-Abfragen eingegeben und deren Ergebnisse dargestellt (Abb. 4-3 unten). Die registrierten Namensräume können auf Wunsch ebenfalls direkt bei der Formulierung von SPARQL-Abfragen berücksichtigt werden (im Bild `rdf` und `db`), indem für diese die registrierten Namensräume automatisch als PREFIX-Ausdruck [PS08, Kap. 4.1.1] vorangestellt werden. So können durch einen Anwender Anfragen deutlich kompakter formuliert werden. Weiterhin können in diesem Fenster formulierte Abfragen direkt zum Anlegen einer neuen Ebene verwendet werden.

Abbildung 4-3 zeigt den Dialog zum Einrichten einer Ebene. Weiterhin wird im Hintergrund in dem linken Teilfenster der Applikation eine Übersicht über die derzeit definierten Ebenen dargestellt. Durch das Hinzufügen einer Ebene mit Reasoning-Unterstützung kann auf Wunsch das Reasoning für die geladenen Ontologien aktiviert werden. Die Abhängigkeiten der Ebenen von der Basis-Ebene und die geladenen Modelle werden als Graph dargestellt (vgl. Abschnitt 3.2.3). Die geladenen Teilmodelle werden mit der gleichen Farbe wie die Basis-Ebene dargestellt.

Abbildung 4-4 zeigt die Umsetzung der Graphdarstellung von RDF-Inhalten mit Ebenenfärbung (vgl. Abschnitt 3.4.2). Um die Übersichtlichkeit zu wahren, werden nur die Elemente dargestellt, welche sich in dem vorgegebenen Radius eines ausgewählten Elementes befinden. Durch Klicken auf ein Element im Graphen wird der Fokus auf dieses Element verschoben. Weiterhin beeinflussen diverse Optionen die Graphdarstellung und der Graph kann exportiert werden. Dennoch ist gerade Darstellung, Layout und Interaktion mit Graphen eine sehr herausfordernde

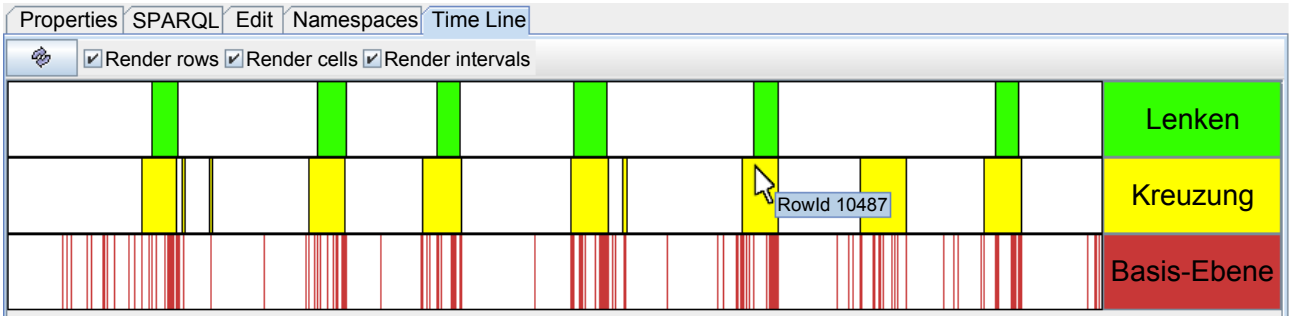


Abbildung 4-5: SDB, Belegungsvisualisierung[2,4] mit Ereignissen aus Abb. 3-17 (nach [NBK11])

Aufgabe, sodass die derzeitige Implementierung primär zur Demonstration für die Integration des Ebenenkonzeptes dient. Als eine der leistungsfähigsten Anwendungen aus dem Bereich der RDF-Graph-Visualisierung kann *Jambalaya* [LS05] gelten.

Die Implementierung der Belegungsvisualisierung aus Abschnitt 3.4.3 und 3.5.3.3 wird in Abb. 4-5 anhand von Ereignissen aus dem Beispielautomaten in Abb. 3-17 dargestellt. Die Visualisierung wird bei der Darstellung so skaliert, dass jeweils die ganze Zeitreihe sichtbar ist. Jede Ebene wird über eine eigene Visualisierungskomponente dargestellt. Weiterhin ist sichtbar, dass in der Basis-Ebene Zeitbereiche noch nicht geschlossen dargestellt werden. Erst die anderen Ebenen setzen eine SPARQL-Abfrage zur Auswahl der Kategorien und ein Reasoning ein, sodass die entsprechenden Annotationen für diese generiert werden (Abb. D-2).

Als weiteres Leistungsmerkmal kann der Semantic Database Browser Tabellen zusammen mit ihren Annotationen in CSV-Dateien exportieren. Je Ebene und Zeile bzw. Zelle wird beim Export eine neue Spalte eingeführt, welche die Annotationen aufnimmt. Über Optionen können mehrere Spalten zusammengefasst oder nur die Anzahl der vorhandenen Annotationen ausgegeben werden. Die Tabellen D-2 und D-3 im Anhang zeigen detailliert aufgeschlüsselt den Umfang des Quellcodes, den der entstandene Demonstrator umfasst.

Während der Arbeit an der Anwendung wurden mehrfach Aktualisierungen am eingesetzten Semantic Web Framework oder dem optionalen Reasoner berücksichtigt, da diese ständig weiterentwickelt werden. Erwähnenswert in diesem Zusammenhang ist, dass das Einladen der NASA-QUDT-Ontologie (Quantities, Units, Dimensions and Data Types, [MHK10]) sich dramatisch beschleunigt hat. Pellet 2.1.1 hatte diese umfangreiche Ontologie nach 6 Stunden nicht geladen, während Pellet 2.2.1 in weniger als 3 Minuten fertig wurde. Solche Fortschritte sind möglich, da derzeit im Bereich des Semantic Web intensiv gearbeitet und geforscht wird, und lassen für die Zukunft weitere Verbesserungen erwarten.

4.3 Anwendung auf SQL-Ebene

Wie in Abschnitt 3.2.2 zur Architektur angesprochen, können die in der Datenbank gespeicherten Annotationen direkt in der Datenbank mittels SQL verarbeitet werden. Im Folgenden

```
1  INSERT INTO mdl_11_20100716_105523_ex_tpl (triple)
2  SELECT
3      SDO_RDF_TRIPLE_S(
4          getmodeltable(
5              mapdbelement('U_MYUSER', '20100716_105523_expstate') || '#bulk-model'
6          ),
7          '<' || mapdbelement(
8              'U_MYUSER',
9              '20100716_105523_expstate',
10             null,
11             a.row_id
12         ) || '>',
13         '<http://ts.dlr.de/ontology/positivesEreignis>',
14         '<http://ts.dlr.de/ontology/Kreuzung>'
15     )
16 FROM "20100716_105523_expstate" a
17 JOIN "20100716_105523_expstate" b
18 ON (a.row_id = b.row_id - 1)
19 WHERE a.CROSSING=1 AND b.CROSSING=0
```

Listing 4-1: Erzeugung von Annotationen mittels SQL

wird erläutert, wie die semantischen Annotationen in einer relationalen Form dargestellt und verarbeitet werden können. Zu berücksichtigen ist, dass die Verarbeitung mittels SQL nicht so komfortabel und anschaulich wie mittels SIMDa-Framework und Semantic Database Browser sein kann, da diese eine deutlich reichhaltigere Abstraktion bereitstellen (Abb. 4-1 und 3-7).

Der Fokus liegt im Folgenden auf Annotation für Massendaten (Zeilen und Zellen), da deren Verarbeitung in relationalen Datenbanken typischerweise im Vordergrund steht. Die Arbeit mit Annotationen für Datenbank-Schema-Objekte erfolgt weitestgehend analog. Die im Folgenden betrachtete Implementierung wird für eine Oracle-Datenbank umgesetzt und nutzt somit die dort vorhandenen nativen Funktionalitäten zur Behandlung von RDF (vgl.a. [MAD+05]). Insbesondere existiert ein Datentyp `SDO_RDF_TRIPLE_S`, der RDF-Aussagen speichert. Eine identische Umsetzung in anderen Datenbankmanagementsystemen ist natürlich ebenfalls möglich. Jedoch müssten für eine alternative Umsetzung verwendete für Oracle spezifische Funktionen nachprogrammiert werden oder etwas anders ausgedrückt werden. Der Grund für diese Notwendigkeit ist, dass die RDF-Funktionalitäten für Datenbanken bisher nicht standardisiert sind.

Aufbauend auf den durch die Datenbank bereitgestellten Funktionalitäten existieren einige im Rahmen dieser Arbeit entwickelte Funktionen in *Procedural Language/SQL* (PL/SQL) zur Ausführung in der Datenbank (Abschnitt D.16). Die Funktion `mapdbelement` erlaubt das Abbilden von Datenbank-Elementen in URIs, sodass diese in RDF-Ausdrücken eingesetzt werden können (Listing D-4). Die Funktion wird in Listing 4-1 eingesetzt, welches die Erzeugung von Annotationen auf SQL-Ebene demonstriert. In diesem Beispiel werden Zeilen entsprechend dem Beispiel aus Abb. 3-18 mit dem Prädikat `positivesEreignis` und dem Objekt `Kreuzung` annotiert, wenn das betrachtete Fahrzeug in eine Kreuzung hineingefahren ist. Dieses Ereignis wird genau dann erkannt, wenn die Variable `CROSSING` der betrachteten Messdatentabelle von 0 auf 1 springt. Die Spalte `row_id` enthält für diese Tabelle eine laufende, eindeutige Zeilennummer.

Die Speicherung der eigentlichen Tripel erfolgt in dem Beispiel in der Annotationen-Tabelle `mdl_11_20100716_105523_ex_tpl`, welche die Massendaten-Annotationen für die Tabelle

```
1 SELECT d.*, a.COLUMN_ID, a.triple.get_triple()
2 FROM "20100716_105523_expstate" d
3 LEFT JOIN table (bulk_annotation('U_MYUSER', '20100716_105523_expstate')) a
4 ON (d.row_id = a.row_id)
5 WHERE a.COLUMN_ID='CROSSING'
6 ORDER BY d.row_id
```

Listing 4-2: Auslesen von Annotationen mittels SQL

`20100716_105523_expstate` aufnimmt. Die Annotationen-Tabelle entspricht somit dem Graph-Modell $G_{T\_D/S/T}$ bzw. $G_{T\_orcl/U\_MYUSER/20100716\_105523\_expstate}$ aus Abb. 3-6. Die Funktion `getmodeltable` wird eingesetzt, um genau dieses Modell bzw. die Tabelle zu bestimmen. Sollen andere Annotationen erstellt werden, müssen lediglich die `WHERE`-Bedingung des Listings und die Annotation selber entsprechend angepasst werden.

Die inverse Aufgabe zum Annotieren stellt das Auslesen der Annotationen mit relationalen Mitteln dar. Hierfür kann die entwickelte Funktion `bulk_annotation` eingesetzt werden (Listing D-6). Als Argument wird ihr eine Tabelle mit Schema übergeben und als Ergebnis liefert sie für diese Tabelle eine neue Tabelle, welche die Massendaten-Annotationen mit der Bezugs-Zeile und Spalte enthält. Die Funktion bestimmt zunächst das entsprechende Modell $G_{T\_D/S/T}$ und durchsucht die zugeordnete Tabelle nach Tripeln mit Subjekten, welche sich auf Zeilen und Zellen beziehen anhand ihrer URI (vgl. Abschnitt 3.2.1). Des Weiteren übernimmt die Funktion die Bestimmung der entsprechenden Spalte und Zeile und liefert diese zusammen mit der Aussage als Ergebnis. Das Listing 4-2 zeigt die beispielhafte Anwendung der Funktion.

In dem Listing werden die Massendaten aus der Tabelle `20100716_105523_expstate` über ein relationales `LEFT JOIN` mit ihren Annotationen verknüpft, sodass in der Ergebnistabelle die Annotationen direkt neben den Massendaten stehen. Die Verknüpfung erfolgt über das Attribut `row_id`, welches in beiden Eingangstabellen vorhanden ist. In dem Beispiel wird das Ergebnis weiterhin nach der Zeilennummer sortiert und es erfolgt eine Filterung über eine `WHERE`-Bedingung, sodass nur Annotationen für die `CROSSING`-Spalte angezeigt werden. Entfällt der Einsatz des optionalen Filters, würden sämtliche Annotationen zurückgeliefert werden, wobei Zeilen-Annotationen dadurch identifiziert werden, dass die Spalte `COLUMN_ID` keine Angabe bzw. `NULL` enthält.

Wie in den letzten Abschnitten demonstriert wurde, ist die Verknüpfung von Tabellen und semantischen Annotationen auf relationaler Ebene möglich und einfach anzuwenden. Auch wenn in der Datenbank sogar Reasoning-Mechanismen angeboten werden, reichen diese in der Funktionalität derzeit nicht an einen spezialisierten Reasoner heran. Weiterhin können Annotationen auf diese Weise relational verarbeitet werden, jedoch wird so nicht von den im letzten Abschnitt beschriebenen Abstraktions- und Interaktionskonzepten profitiert. So ist die Verarbeitung auf SQL-Ebene als Ergänzung zu den behandelten Konzepten zu verstehen.

4.4 Zusammenfassung und Bewertung

4.4.1 Zusammenfassung

In diesem Kapitel wird die Umsetzbarkeit der in Abschnitt 3 eingeführten Konzepte demonstriert. So wird zunächst das SIMDa-Framework eingeführt, welches wesentliche Elemente der zuvor behandelten Architektur implementiert und somit durch Applikationen entsprechend verwendet werden kann. Der Semantic Database Browser realisiert auf dieser Basis eine Applikation, um semantische Datenbank-Annotationen zu visualisieren und um mit ihnen zu interagieren.

4.4.2 Bewertung

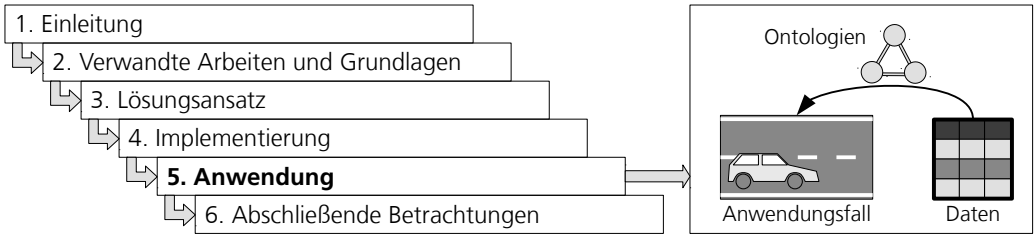
Die Implementierung zeigt, dass die dieser Arbeit zugrunde liegende Vision zur Kombination einer relationalen Datenbank mit Semantic Web Technologien realisierbar und nutzbar ist.

Die hergeleitete Architektur hat sich als praktisch umsetzbar und gut einsetzbar erwiesen. Das SIMDa-Framework bildet eine Grundlage, die neben dem Einsatz im Semantic Database Browser in weiteren Applikationen zur Arbeit auf der gemeinsamen Datenbasis dienen kann. Im Architekturabschnitt eingeführte allgemeine Funktionalitäten werden im SIMDa-Framework angesiedelt. Durch die entsprechende Gesamtarchitektur werden bei der Implementierung bereits existierende Komponenten und Bibliotheken wiederverwendet, sodass sich die Umsetzung voll auf die Konzepte dieser Arbeit fokussiert.

Im Semantic Database Browser werden die Konzepte der Vision, welche dieser Arbeit zugrunde liegen, so realisiert, dass sie adäquat veranschaulicht werden und sich gut nachvollziehen lassen. Ebenso wird im Semantic Database Browser deutlich, dass sich die verschieden eingeführten Visualisierungsvarianten zur interaktiven Exploration der Datenbestände nutzen lassen. Dieser Umstand trifft insbesondere auf die Visualisierung von Ebenen zur Einfärbung von Tabellen und in der Belegungsvisualisierung zu.

Auch wenn es sich bei der vorliegenden Implementierung um eine prototypische Umsetzung handelt, zeigt sie bereits, dass die erarbeiteten Konzepte praktisch umsetzbar und anwendbar sind. Die in den Grundlagenkapiteln aufgezeigten Schwächen verwandter Arbeiten werden durch die vorgestellte Implementierung geschlossen und das Management von Versuchsdaten um neuartige Möglichkeiten wesentlich erweitert. Dabei erfolgt noch keine Einschränkung in der Implementierung auf die Anwendung für bestimmte Domänen, sodass die Implementierung generisch eingesetzt werden kann. Das nächste Kapitel zeigt eine exemplarische Anwendung auf die Domäne von Versuchsfahrten und Fahrmanövern.

5 Anwendung



In den bisherigen Kapiteln wurden die Konzepte und eine Implementierung zum *Einsatz semantischer Technologien zur formalen Beschreibung von Zeitreihen in Datenbanken* eingeführt. Das folgende Kapitel zeigt aufbauend auf diesen Grundlagen, wie konkrete *domänenspezifische* Ontologien zur Beschreibung von Zeitreihen eingesetzt werden können.

5.1 QUDT-Ontologie

Die Ontologie *Quantities, Units, Dimensions and Data Types in OWL and XML* (QUDT, [MHK10]) der NASA beschreibt umfassend physikalische Einheiten, Dimensionen und deren Zusammenhänge. Sie umfasst ca. 2 MB aufgeteilt in mehrere OWL-Dateien. Die Verwendung dieser Ontologie bietet sich somit an, Spalten einer Tabelle, welche Messwerte enthält, mit der entsprechenden physikalischen Einheit zu annotieren, in denen die Messwerte vorliegen.

Die QUDT-Ontologie demonstriert anschaulich die *Wiederverwendbarkeit* und *Verlinkung von Ressourcen*. Die Wiederverwendbarkeit ergibt sich daraus, dass ein umfangreiches Modell direkt integriert werden kann und somit nicht selbständig mit viel Aufwand entwickelt werden muss. Über eine physikalische Einheit kann in der Ontologie zu einer entsprechenden Ressource in der DBpedia navigiert werden (vgl. Abschnitt 2.3.7). Über die DBpedia können Beziehungen zu vielen weiteren Ontologien abgeleitet werden (Abb. 2-16). Weiterhin enthält die QUDT-Ontologie Verweise auf die entsprechenden Seiten der Wikipedia, dass menschenverständliche Erklärungen direkt annavigiert werden können, was z.B. direkt in der Baumdarstellung des Semantic Database Browser erfolgen kann. (Im Anhang stellen Abb. D-4 und Listing D-7 diese vorhandenen Beziehungen explizit dar.) Aus diesen Tatsachen folgt die *Verknüpfung mit (fremden) Ressourcen*.

5.2 Datenbank-Ontologie

Die Erzeugung und der Einsatz semantischer Annotationen für Datenbank-Elemente erlaubt die Referenzierung dieser Elemente mit semantischen Technologien (vgl. Abschnitt 3.2.1). Dennoch ist in einer auf Ontologien basierenden Wissensdarstellung die Suche oder die Identifizierung von diesen Elementen mittels entsprechender Konzepte erst einmal nicht möglich. Dies wird erst dann möglich, wenn eine RDF-Ressource, welche z.B. eine Tabelle repräsentiert, als Instanz eines Tabellen-Konzeptes explizit ausgewiesen wird. Aus diesem Grund wird bei Bedarf eine *Datenbank-Ontologie* eingesetzt, welche die grundlegenden Konzepte einer Datenbank erfasst und auch im Anhang in Abb. D-5 dargestellt ist.

Mit SPARQL können so explizit Tabellen oder andere Datenbank-Elemente aufgrund ihres Typs oder ihrer Beziehungen zueinander abgefragt werden und somit entsprechend für Filter in Ebenen eingesetzt werden. Weiterhin wird mittels des Datatype-Attributs `rdfs:label` den Datenbank-Elementen ihr Name zugeordnet, sodass dieser explizit abfragbar ist und nicht zwangsweise durch einen menschlichen Anwender aus der entsprechenden URI herausgelesen werden muss.

Die Verwendung der Datenbank-Ontologie führt zu einem weiteren positiven Effekt. Werden sämtliche annotierte Datenbank-Elemente ihren jeweils enthaltenden Elementen über die Ontologie zugeordnet, kann ausgehend von dem eine Datenbank repräsentierenden Element, welches als Wurzel dient, zu den enthaltenen Schemas, Tabellen etc. navigiert werden. Dies führt insbesondere beim Einsatz des Semantic Database Browser dazu, dass in der Baumansicht der Datenbank ein Schema oder eine Tabelle gefärbt dargestellt wird, wenn dort annotierte Spalten, Zeilen oder Zellen existieren. Somit kann ein Anwender schon früh erkennen, welche Bereiche der Datenbank Annotationen besitzen.

5.3 Ontologie für Fahrereignisse und Manöver

Die im Folgenden beschriebene und im Rahmen dieser Arbeit entstandene Ontologie für Fahrereignisse und Manöver ist ein Beispiel für eine domänenspezifische Ontologie, da sie einen eher speziellen Fokus besitzt. Sie demonstriert somit, wie die in dieser Arbeit eingeführten Prinzipien sich für *domänenspezifische Aufgaben* nutzen lassen. Weiterhin ist die vorgestellte Ontologie so ausgestaltet, dass gezeigt wird, dass *Reasoning* sinnvoll eingesetzt werden kann. Für diese Ontologie wird außerdem ein semantisches Wiki mit integriert, sodass weitergehende Aspekte zur *Einbeziehung externer Ressourcen* und eines *verteilten Systems* mit betrachtet werden.

Inhaltlich beruht die Ontologie auf [VSA+05, Sch09a]. Sie beschreibt eine Menge von 15 Fahrmanövern in Anlehnung an [Nag94]. Weiterhin sind in ihr 28 Ereignisse bzw. Aktionen erfasst, welche während einer Messfahrt detektiert werden können. Sie umfassen Elemente der Fahrzeugbedienung, Fahrzeugreaktionen, voranfahrende Fahrzeuge, die Position des Fahrzeuges

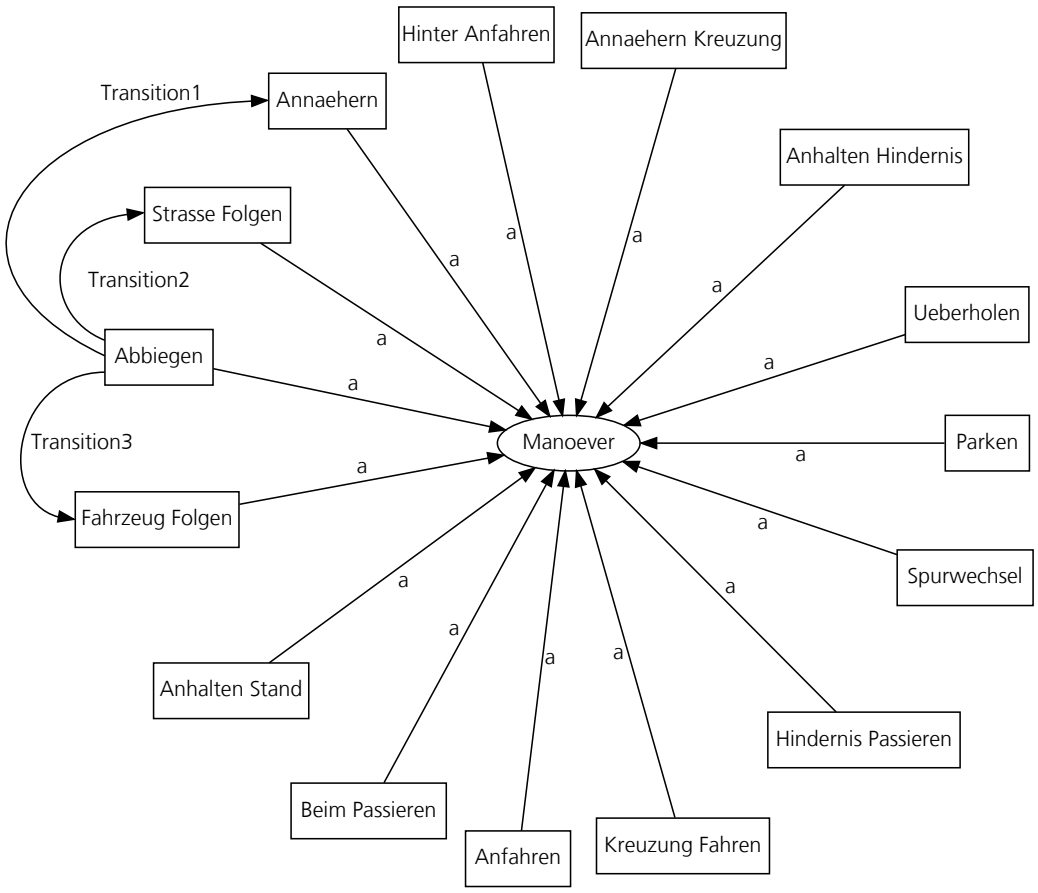


Abbildung 5-1: Instanzen der Manöver-Klasse und Transitionen für `Abbiegen`

auf der Fahrspur und der derzeitig befahrenen Strecke. Diese Informationen können vom *Controller Area Network Bus* (CAN-Bus) eines Fahrzeuges oder von in einem Serienfahrzeug verbauten Sensoren mit digitaler Karte entnommen werden. Basierend auf den Ereignissen werden die Fahrmanöver beschrieben. Die Ereignisse dieser Ontologie entsprechen somit den in Abschnitt 3.5 eingeführten *Elementarereignissen* und werden während einer Analyse der Messdaten als semantische Annotationen der Zeitreihen erzeugt. Die Fahrmanöver dagegen werden in der Ontologie als komplexe Zusammenhänge auf Basis der Ereignisse beschrieben. Abbildung 5-1 zeigt die Individuen der Manöver-Ontologie, welche die Fahrmanöver repräsentieren. Das in der Abbildung verwendete Attribut `a` ist eine übliche Abkürzung für `rdf:type`.

Die Beschreibung der Beziehungen zwischen den Ereignissen und den Manövern erfolgt mittels eines deterministischen endlichen Automaten in der Ontologie, wie er in Abschnitt 3.5.2 und 3.5.3 beschrieben ist. Abweichend zu den in Abschnitt 3.5.3.2 eingeführten Modellierungsdetails wird eine erweiterte Modellierung zur Berücksichtigung vorhergehender Ereignisse durch Reasoning eingesetzt (Abb. D-6). Die zusätzlich eingeführten Sub-Properties erlauben es zu erkennen, welche Ereignisse explizit annotiert wurden und welche durch Reasoning von vorhergehenden Annotationen abgeleitet wurden. Als Startzustand q_0 des Automaten dient das Manöver `Anhalten Stand`, welches das Stehen des Fahrzeuges an einer Stelle repräsentiert, so wie es zu Beginn einer Fahrt zwangsweise auftritt.

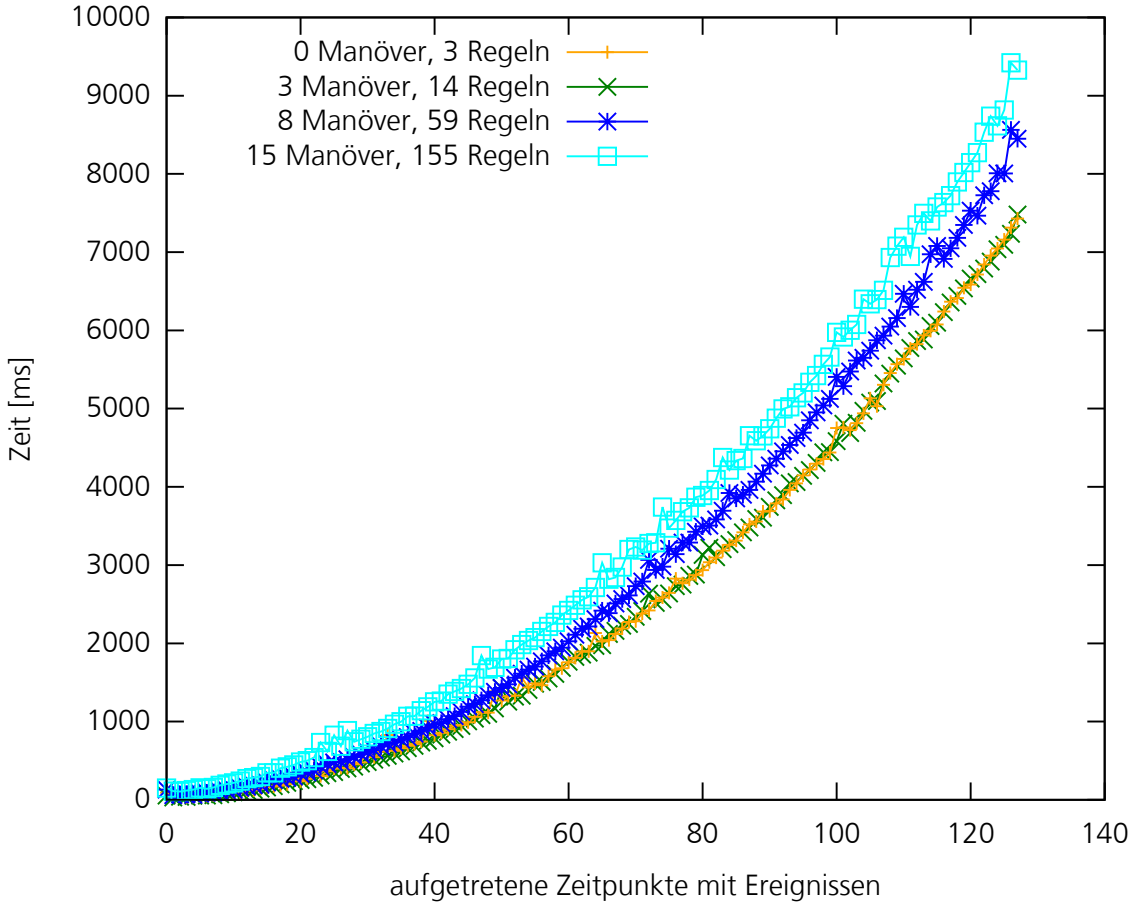


Abbildung 5-2: Zeit zur Auswertung der Manöver-Ontologie

Die so realisierte Manöver-Ontologie umfasst 155 Regelausdrücke für den Automaten bestehend aus 1104 Atomen. In Abb. 5-1 sind ausschließlich die möglichen Transitionen für das Manöver `Abbiegen` eingezeichnet. Es ist leicht zu erahnen, dass sämtliche 155 Transitionen des Automaten für einen menschlichen Betrachter nur schwer zu erfassen wären. Die Individuen werden einer Klassenhierarchie von 33 Klassen zugeordnet. Es existieren 49 Individuen und eine weitere für jeden zu annotierenden Zeitpunkt bzw. Zeile. Von diesen Individuen repräsentieren 28 Ereignisse und 15 Manöver.

5.3.1 Betrachtungen zur Laufzeit für Fahrereignisse und Manöver

Abbildung 5-2 zeigt die benötigte Zeit zur Auswertung der Manöver-Ontologie mit einem Reasoner, bei der der Automat berechnet wird, in Abhängigkeit von der Anzahl annotierter Zeilen. Es sind jeweils 4 Konfigurationen geplottet, welche Abstufungen des Automaten bilden, bei der nur jeweils eine Teilmenge der Zustände bzw. Manöver vorhanden sind. Es ist zu erkennen, dass bei den betrachteten Größenordnungen die Größe des Automaten nur bedingten Einfluss auf die Laufzeit besitzt, während eine Erhöhung der annotierten Zeilen deutlich merkbar ist. Für den vollständigen Automaten mit 125 annotierten Zeilen werden zur Auswertung ca. 8,8 Sekunden benötigt.

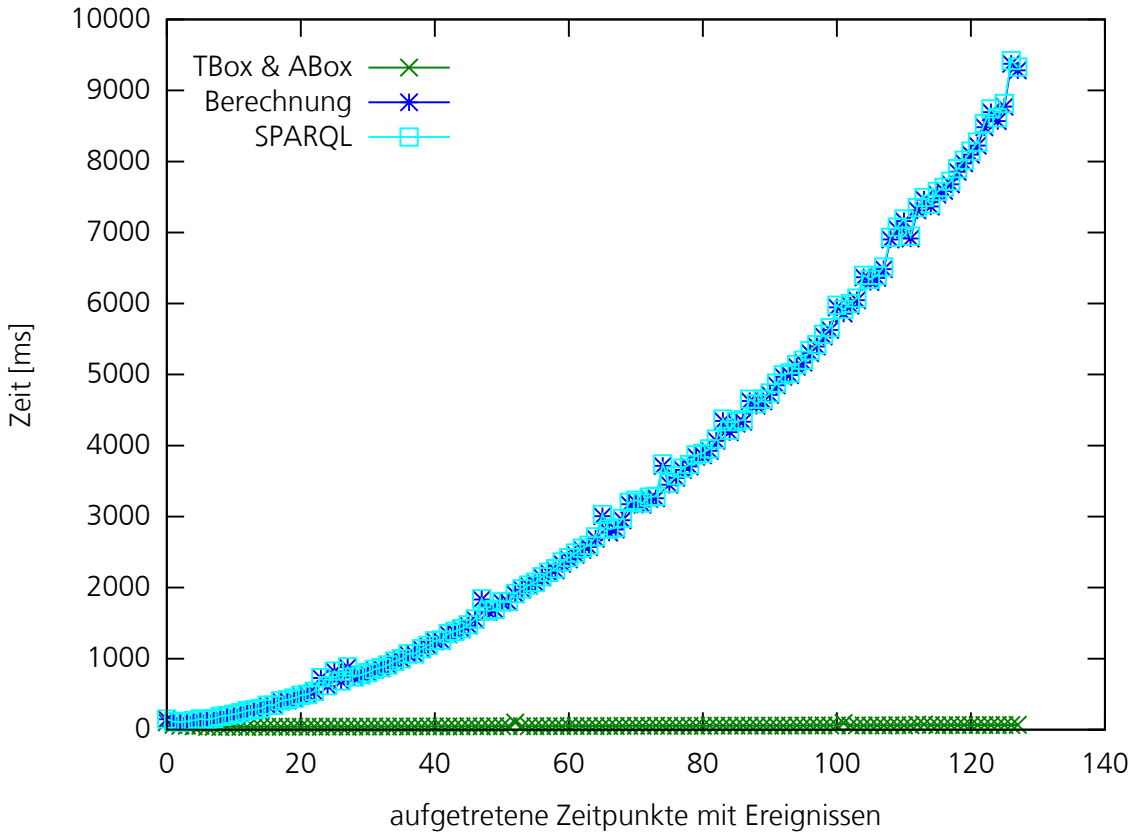


Abbildung 5-3: Aufgeschlüsselte Zeiten für 15 Manöver

Die zeitliche Häufigkeit von Ereignissen ist stark abhängig vom betrachteten Szenario. Während in städtischen Umgebungen Ereignisse vergleichsweise häufig auftreten können, ist die Häufigkeit auf Überlandfahrten als eher gering zu betrachten. Bei einem angenommenen durchschnittlichen Abstand von 10 Sekunden zwischen zwei Ereignissen können mit 125 Zeilen ca. 21 Minuten Fahrzeit beschrieben werden.

In Abb. 5-3 wird die Zeit für die einzelnen Schritte für den vollständigen Automaten weiter aufgeschlüsselt dargestellt, wobei die Zeiten der Schritte kumulativ zu interpretieren sind. Als erster Schritt wird die TBox (Schemawissen) und danach die ABox (Instanzwissen) erstellt. Anschließend erfolgt die eigentliche Berechnung der Ontologie bzw. das Reasoning. Als letzter Schritt der Auswertung werden mittels folgender SPARQL-Abfrage genau die Zeilen mit den berechneten Manövern abgerufen:

```
1 PREFIX maneuver: <http://ts.dlr.de/ontology/>
2 SELECT ?point ?maneuver
3     WHERE { ?point maneuver:maneuver ?maneuver }
4     ORDER BY ?point
```

Wie zu erwarten, dauert die eigentliche Berechnung am längsten. Sowohl das Erstellen der TBox und ABox als auch das Abfragen der Ergebnisse mit SPARQL erfolgen dagegen so schnell, dass sie kaum wahrzunehmen sind. (Im Anhang zeigt Tab. D-4 detailliert die benötigten Zeiten für ausgewählte Zeilenanzahlen.) Diese Messungen wurden auf einem Intel Core 2 Duo E3500 mit

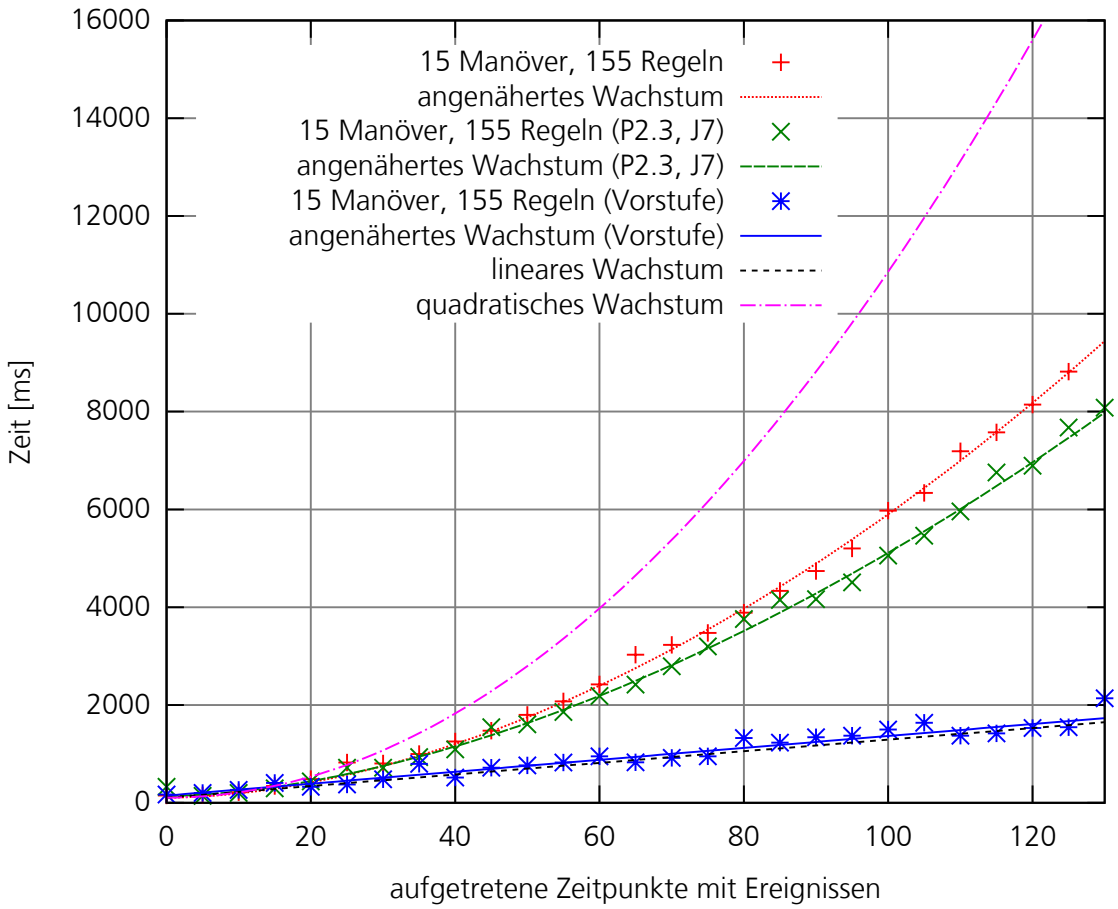
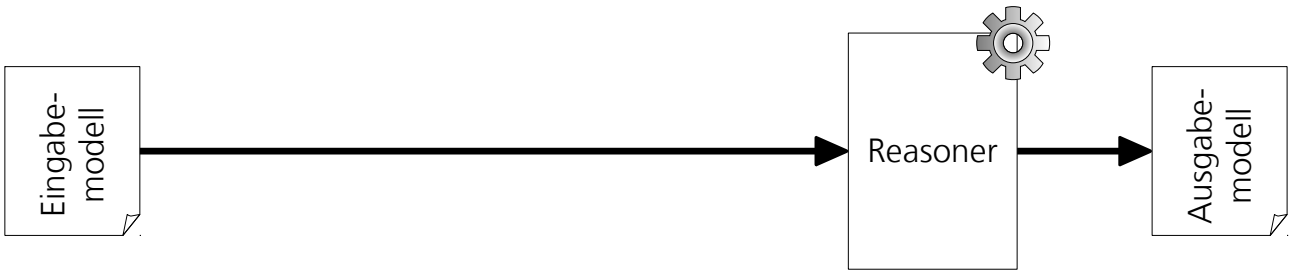


Abbildung 5-4: Zeit zur Auswertung der Manöver-Ontologie im Vergleich mit verschiedenen Funktionen

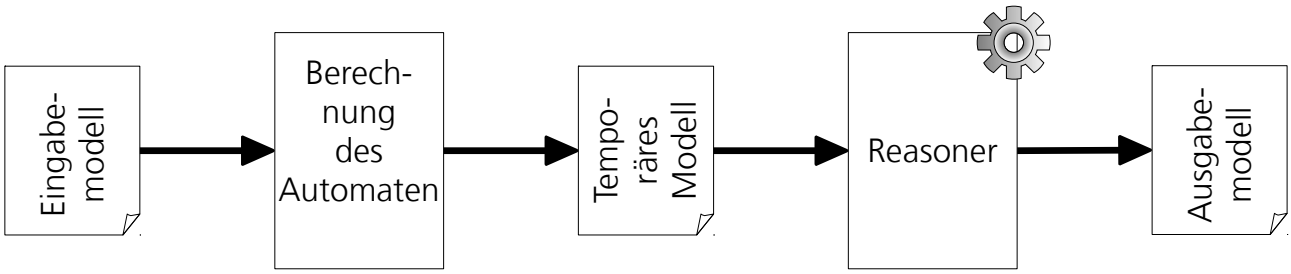
3,16 GHz und 4 GB Hauptspeicher auf Windows 7 durchgeführt. Java wurde in der Version 6 Update 22 und Pellet in der Version 2.2.2 eingesetzt.

In Abb. 5-4 wird die benötigte Zeit zur Auswertung der Manöver-Ontologie aus Abb. 5-2 im Vergleich zu einigen anderen Funktionen dargestellt. Die erfassten Messwerte, von denen zur besseren Übersichtlichkeit nur jeder fünfte dargestellt ist, werden durch eine Funktion $f(x) = 105 + b \cdot x^a$ angenähert. Der Y-Achsenabschnitt der Funktion beträgt 105, da bei den Messwerten mindestens 105 ms für die Berechnung benötigt wurden. Die Parameter $a \approx 1.8$ und $b \approx 1.3$ sind so gewählt, dass die Messwerte optimal angenähert werden. Somit dient f zur Abschätzung der konkreten Laufzeitkomplexität für das eingesetzte Reasoning (vgl. Abschnitt 3.5.4).

Zum Vergleich werden eine lineare und eine quadratische Funktion dargestellt. Diese sind so parametrisiert, dass der Fehler für die ersten 15 Messwerte minimiert wurde (Abschnitt D.17) und der weitere Funktionsverlauf eine fiktive Laufzeit extrapoliert. Diese Funktionen stellen somit eine Abschätzung dar, wie sich die Laufzeit bei linearer oder quadratischer Komplexität verhalten würde. Ein lineares Verhalten ist das theoretische Optimum. Im Verlauf dieser Arbeiten wurden die durchgeführten Messungen mit den aktualisierten Versionen Pellet 2.3 und Java 7 (P2.3, J7) wiederholt, welche ebenfalls zum Vergleich dargestellt sind und zeigen, dass sich die eingesetzten Implementierungen bereits verbessert haben. Der mit „Vorstufe“ gekennzeichnete



(a) Direkte Berechnung eines Modells mit einem Reasoner



(b) Auswertung eines Modells mit zuvoriger Berechnung des Automaten in einer Vorstufe

Abbildung 5-5: Alternativen zur Berechnung eines Automaten

Graph wird im nächsten Absatz erläutert. Sämtliche dieser genannten Komplexitäten gehören zur Komplexitätsklasse $\mathcal{P}$ (lösbar in Polynomialzeit) und noch nicht zur Komplexitätsklasse $\mathcal{NP}$ (nichtdeterministisch polynomielle Zeit) (vgl.a. [CLR01, Kap. 36, S. 916 ff.]). Folglich können sie alle als praktisch lösbar betrachtet werden.

Ein pragmatischer Ansatz zur Verbesserung der Rechenzeit ist in Abb. 5-5 dargestellt. Anstatt ein Modell direkt zur Berechnung an einen Reasoner zu übergeben, kann in einem Vorverarbeitungsschritt zunächst ein enthaltener Automat berechnet werden. Diese Berechnung kann auf die Ausführung von Automaten optimiert werden und so für diesen Schritt eine lineare Zeitkomplexität sichergestellt werden. Diese Vorgehensweise ist möglich, da beim *monotonen* Reasoning im Semantic Web nach der *Open World Assumption* einem Modell nur neue Aussagen als Schlussfolgerung hinzugefügt werden, aber nicht entfernt werden (vgl. [GM05]). Dadurch müssen im finalen, regulären Reasoning-Schritt weniger Berechnungen durchgeführt werden. Die auf diese Art erzielten Messergebnisse sind in Abb. 5-4 mit der Bezeichnung „Vorstufe“ gekennzeichnet. Dort ist zu sehen, dass mit der Vorstufe zur Berechnung des Automaten im betrachteten Bereich eine näherungsweise lineare Komplexität erreicht wird, welche nur geringfügig schlechter als die in dieser Abbildung gezeigte lineare Extrapolation ist.

Zum Vergleich wurde dieselbe Ontologie mit 125 Ereignissen in Protégé geladen, um sie dort mit vergleichbaren Reasonern als Benchmark zu berechnen (vgl.a. [GHT06, BHJV08]). Hermit 1.3.5 [MSH09] hat zur Berechnung ca. die 13-fache Zeit von Pellet benötigt. Fact++ [TH06] in der Version 1.5.2 war zwar bereits nach beeindruckenden 156 ms fertig, hat jedoch die SWRL-Ausdrücke nicht korrekt ausgewertet, sodass am Ende die Manöver nicht berechnet waren. Bei RacerPro [HHMW12] handelt es sich um ein kommerzielles Produkt, sodass lediglich die öffentliche Preview der noch nicht fertiggestellten Version 2 getestet werden konnte, welche

Abbildung 5-6: Auszug aus semantischem Wiki zu dem Manöver *Anhalten Stand*[4]

jedoch nicht in der Lage war, die betrachtete Ontologie ohne Fehler zu laden. In ähnlichem Zusammenhang wird auch der Reasoner Kaon2 [Mot06, Kap. 12, S. 203 ff.] genannt, dessen letzte Version stammt jedoch von Juni 2008 und kann somit als nicht mehr aktuell betrachtet werden.

Bezüglich des Automaten und dessen Transitionen ist hervorzuheben, dass obwohl diese berechnet werden, sie ebenso ein Teil der Ontologie sind und wie die Ereignisse und Manöver behandelt werden können. So lassen sich mittels Abfragen Teile des Automaten extrahieren, um diese weiterzuverarbeiten, z.B. als Diagramme visualisieren. Erfolgt dagegen eine Umsetzung in einer klassischen Programmiersprache (z.B. C++ oder Java), wird zwischen der Programmlogik und den Daten (Eingabe und Ausgabe) strikt getrennt, sodass eine solche Verwendung nicht möglich ist.

5.3.2 Anbindung der Manöver-Ontologie an ein semantisches Wiki

Die mittels Reasoning erzielten Ergebnisse lassen sich genau wie nicht abgeleitete Annotationen im Semantic Database Browser betrachten. Im Falle der betrachteten Manöver bietet sich eine Visualisierung in der Belegungsvisualisierung an (Abb. 4-5). Im Semantic Database Browser wird die *Einbeziehung externer Ressourcen* und Aspekte eines *verteilten Systems* unter Einbeziehung eines semantischen Wikis demonstriert. Die URI eines Manövers ist in diesem Fall so gewählt, dass sich hinter ihr eine beschreibende Webseite verbirgt (Abb. 5-6), und entspricht der externen Ressource. Bei dieser Webseite handelt es sich um eine Seite des semantischen Wikis, in dem eine ausführliche menschenverständliche Beschreibung des Manövers vorgenommen wird.

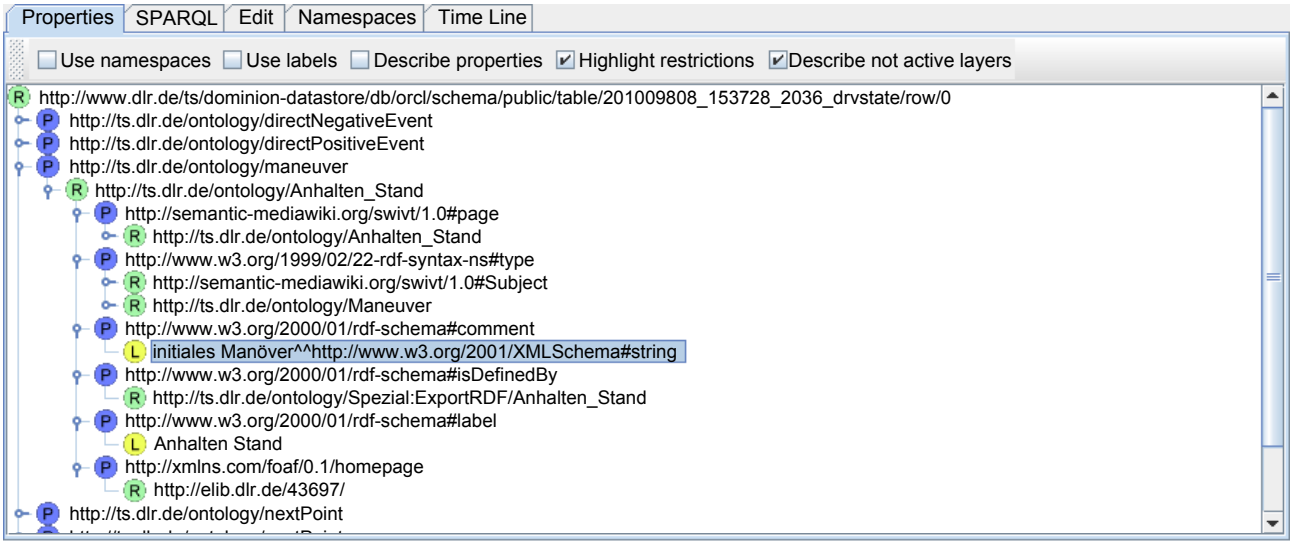


Abbildung 5-7: Annotationen aus dem semantischen Wiki im Annotationsbaum des SDB[4]

Durch den Wiki-Charakter dieser Seiten lassen sich durch Anwender diese Informationen leicht erweitern oder neue Einträge für andere Elemente anlegen.

Da es sich um ein semantisches Wiki handelt, werden Informationen nicht nur als formatierter Text dargestellt, sondern können als formale Tripel ausgedrückt werden. Diese werden zum einen unten auf der Seite in der zusammenfassenden Fakten-Box dargestellt, können aber auch ausschnittsweise oder vollständig in RDF/XML-Syntax abgerufen werden (Listing D-8). Die RDF-Darstellung zu der aktuellen Seite wird über den Link *RDF-Feed* der Fakten-Box abgerufen. Eine solche Darstellung lässt sich wie jedes andere Modell in den Semantic Database Browser einlesen und entsprechend verarbeiten. Abbildung 5-7 zeigt die in den Annotationsbaum des Semantic Database Browser eingebunden Fakten aus dem Wiki. Zusätzlich zu den dort explizit angegebenen Annotationen werden noch automatisch durch das Wiki generierte Annotationen dargestellt, welche den Ursprung der Annotationen beschreiben. Bei dieser dynamischen Einbindung der externen Wissensrepräsentation bilden der Semantic Database Browser mit der Datenbank und das Wiki ein verteiltes System (vgl. Abschnitt 3.2.4).

5.4 Zusammenfassung und Bewertung

5.4.1 Zusammenfassung

Anhand konkreter Ontologien werden beispielhaft semantische Annotationen für eine Domäne umgesetzt. Diese demonstrieren die praktische Anwendung semantischer Annotationen, welche z.B. im Semantic Database Browser visualisiert werden können. Die ausgewählten Ontologien zeigen die sich ergebenden Möglichkeiten, Reasoning in Kombination mit den annotierten Datenbank-Elementen einzusetzen. Der Schwerpunkt liegt auf der Verarbeitung von Ereignissen

in Zeitreihen. Weiterhin wird die Einbeziehung externer Ressourcen betrachtet und die Anwendung als Teil eines verteilten Systems.

5.4.2 Bewertung

Die gezeigte Referenzierung externer Ressourcen wie Dateien und Webseiten ist für einen praktischen Einsatz ungemein hilfreich. Insbesondere wenn viele Anwender mit einer gemeinsamen Datenbasis arbeiten, erlaubt eine auf diese Weise durchgeführte Dokumentation, wesentlich das Verständnis zu erhöhen. Hierdurch wird zugleich die Qualität und Wiederverwendbarkeit der gespeicherten Daten nachhaltig verbessert.

Die Möglichkeit, fremde Ontologien aufgrund der offenen Architektur der Semantic Web Technologien direkt mit einbinden zu können, ist ebenfalls ein entscheidender Vorteil, da so unnötige Mehrfachimplementierungen leicht vermieden werden können. Weiterhin wird hierdurch die Arbeitsteilung und Kooperation zwischen Experten gefördert, da sich jeder auf seine Spezialgebiete konzentrieren kann. So können etablierte Methoden und Werkzeuge aus dem Bereich des Wissensmanagements übertragen und angewendet werden. Der Einsatz eines semantischen Wikis zur Beschreibung von Fahrmanövern ist eine solche Lösung, die es erlaubt, eine menschenlesbare Darstellung mit einer formalen Darstellung zu kombinieren und direkt mit einzubinden. Durch die direkte Einbindung der Wissensmodellierung können so Zwischenschritte bei der Integration vermieden werden.

Die Manöver-Ontologie demonstriert erfolgreich die Verarbeitung von Ereignissen mit semantischen Technologien. Die bei der Berechnung durchgeführten Messungen geben einen guten Einblick, dass das Reasoning gefordert wird. In dem betrachteten Umfang lassen sich die Berechnungen in akzeptabler Zeit durchführen, dennoch weist die benötigte Zeitdauer bei steigender Anzahl von Ereignissen ein nahezu quadratisches Wachstum auf. Aus diesem Grund können beim derzeitigen Stand der Technik nicht beliebig lange Testfahrten durchgerechnet werden.

Unterschiedliche Reasoner benötigen verschieden lange zur Berechnung von Problemen (vgl.a. [WLLB06, BHJV08]). Für den vorliegenden Anwendungsfall ist der eingesetzte Reasoner Pellet eine geeignete Wahl. Von diesem Standpunkt aus ist es jedoch wünschenswert, dass sich Reasoner in Zukunft noch weiterentwickeln. Dass die Reasoner noch schneller werden können, wird anhand der eingeführten Vorstufe zur Automatenberechung demonstriert. Weiterhin stellt diese Vorstufe ein pragmatisches Hilfsmittel dar, um die notwendige Rechenzeit deutlich zu reduzieren. Da Reasoning-Techniken ein intensiver Forschungsgegenstand sind, besteht jedoch Grund zum Optimismus für weitere Fortschritte. Auch während der Bearbeitungszeit der vorliegenden Arbeit waren deutliche Verbesserungen zu erfahren.

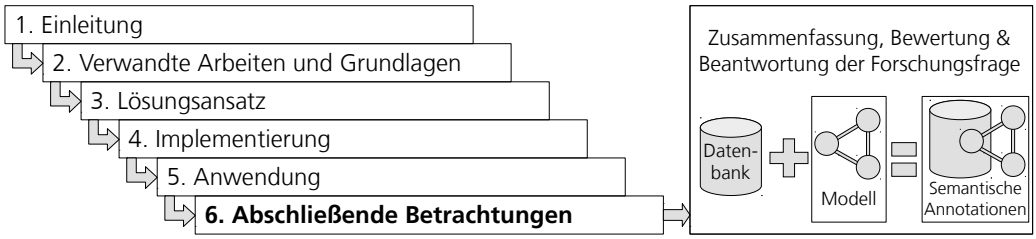
Aufgrund der begrenzten Reasoning-Kapazitäten ist aber eine möglichst kompakte Darstellung der betrachteten Tatsachen umso wichtiger. Eine kompakte Darstellung wird insbesondere

durch zwei Tatsachen in der vorliegenden Arbeit adressiert. Zum einen resultiert aus der Aufteilung der semantischen Annotationen auf mehrere Modelle, dass nur wirklich relevante Annotationen gerade betrachtet werden (vgl. Abschnitt 3.2.3). Zum anderen trägt die Modellierung von Ereignissen in Anlehnung an die semantische Dichte zu einem möglichst minimalen Modell bei (vgl. Abschnitt 3.5.1).

In diesem Abschnitt erfolgt durch den Einsatz domänenspezifischer Ontologien der Bezug zu konkreten Anwendungsdomänen. Durch den Einsatz entsprechender anderer Ontologien kann somit leicht der Bezug zu anderen Domänen hergestellt werden. Die zugrundeliegenden Konzepte und Methoden haben sich daher als generisch und vollständig domänenunabhängig erwiesen. Lediglich die Belegungsvisualisierung und die Beschreibung von Zeitreihen mittels endlicher Automaten benötigen Daten, welche eine natürliche Ordnung besitzen. Alle weiteren Konzepte sind unabhängig von Zeitreihen und können allgemein für beliebige Datenbankinhalte verwendet werden.

Die in diesem Kapitel demonstrierten Anwendungen der eingeführten Konzepte zeigen, dass diese gewinnbringend einsetzbar sind. Für die Anwender entstehen mehrere Vorteile und die Arbeit wird in verschiedenen Aspekten unterstützt. Durch den *Einsatz semantischer Technologien zur domänenspezifischen und formalen Beschreibung von Zeitreihen in Datenbanken* wird das wissenschaftliche Datenmanagement entscheidend verbessert. Im folgenden Kapitel wird die der Arbeit zugrunde liegende Forschungsfrage detailliert zu den einzelnen Aspekten abschließend beantwortet.

6 Abschließende Betrachtungen



In diesem Abschnitt erfolgt die Reflektion der in dieser Arbeit vorgestellten Inhalte. Aufbauend auf einer Zusammenfassung werden die erreichten Ergebnisse in Bezug zur gestellten Forschungsfrage gesetzt. Abschließend erfolgt ein Ausblick auf mögliche zukünftige Fortsetzungen der Arbeit.

6.1 Zusammenfassung und Bewertung

6.1.1 Zusammenfassung

Während der Entwicklung und Erprobung technischer Systeme wird oft verlangt, dass formale Modelle in Beziehung zu Messdaten, welche im Rahmen von Experimenten oder empirischen Studien (z.B. Versuchsfahrten) anfallen, gesetzt werden müssen. Diese Motivation führt zu der Fragestellung, wie formale Modelle für die Beschreibung von Messdaten, welche als Zeitreihen in Datenbanken gespeichert sind, eingesetzt werden können.

In Verwandtschaft zu dieser Fragestellung stehen Arbeiten, welche aus verschiedenen Anwendungsklassen zur Messdatenverarbeitung stammen. Diese zeigen, welches Potential bisher nicht genutzt wird, wenn Messdaten nicht formal auf der Ebene von Messwerten beschrieben werden, da keine Aussagen über deren Inhalte getroffen werden können. Insbesondere semantische Technologien zur Realisierung formaler Modelle in Form von Ontologien bilden eine notwendige technische Grundlage für die Behandlung der Fragestellung und für das Verständnis des darauf aufbauenden Lösungsansatzes. Das Schlussfolgern auf den formalen Ontologien, um implizites Wissen aus den Modellen abzuleiten, wird so möglich und als Reasoning bezeichnet.

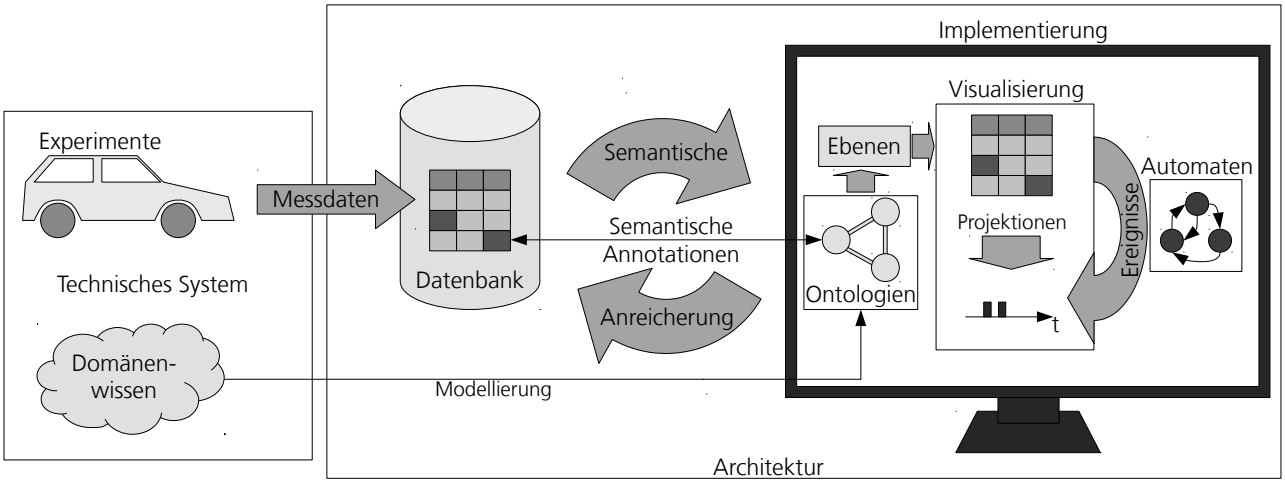


Abbildung 6-1: Übersicht zum Zusammenspiel der verschiedenen Aspekte bei der Beschreibung von Messdaten mit Ontologien

Der Lösungsansatz führt die grundlegende Idee ein, in Form von semantischen Datenbank-Annotationen eine relationale Datenbank mit einem durch Semantic Web Technologien realisierten formalen Modell zu verknüpfen. Eine semantische Annotation ist ein Bestandteil eines formalen Modells, welches eine Aussage über ein Datenbank-Element trifft und somit eine Verknüpfung zwischen den in der Datenbank enthaltenen Messdaten und den formalen Modellen herstellt. Anhand der Kombination von Datenbanken und Semantic Web Technologien ergibt sich die Chance, die Vorteile dieser beiden Technologien miteinander zu kombinieren. Eine geeignete Architektur erlaubt die gemeinsame Datenhaltung, welche die Skalierbarkeit der betrachteten Lösung für große Datenmengen sicherstellt. Der Prozess zur semantischen Anreicherung beschreibt die Verwendung der Annotationen im Rahmen von Datenanalyseanwendungen, sodass ein Abgleich der Modellebene mit den Daten möglich wird. Um die Annotationen handhaben zu können, ermöglichen neuartige Konzepte die Visualisierung und Interaktion mit semantischen Datenbank-Annotationen. Das Konzept der Ebenen erlaubt die gezielte Auswahl und Darstellung ausgewählter Informationen des betrachteten Wissensmodells. Mit Ontologien modellierte generische Automaten erlauben die Anwendbarkeit der Annotationen zur Beschreibung von Zeitreihen und darin auftretenden Ereignissen. Ereignisse in den Messdaten werden als Annotationen umgesetzt und über Reasoning berechnen die Automaten so dynamisch komplexe Ereignisse, was sich am Beispiel von Fahrversuchen demonstrieren lässt.

Eine Implementierung realisiert die eingeführten Konzepte, um so deren Anwendbarkeit zu demonstrieren. Das SIMDa-Framework implementiert die Architekturaspekte, sodass die Konzepte einfacher umgesetzt und besser wiederverwendet werden können. Aufbauend auf das Framework setzt der Semantic Database Browser die eingeführten Visualisierungs- und Interaktionskonzepte um und erlaubt so, semantische Datenbank-Annotationen praktisch für Messdaten anzuwenden.

Als Anwendungsbeispiel für den theoretischen Lösungsansatz veranschaulichen konkrete Ontologien den Einsatz der Annotationen für die Domäne von Versuchsfahrten. Diese Ontologien erlauben die formale Beschreibung von Fahrmanövern als Automaten, welche durch Reasoning

		Informations-systeme für Fahrversuche		Scientific Workflow Systems		Wissenschaftliche Datenhaltungs-systeme	
Einsatzgebiet		Messdaten-visualisierung, Plots, Videos		Modellierung und Ausführung von Workflows		Archivierung von Messdaten	
Semantische Annotationen		ohne	**mit**	ohne	**mit**	ohne	**mit**
Leistungsmerkmale	Datenverwaltung	++	**++**	+	**++**	+	**++**
	Interpretation der Dateninhalte	++	**++**	o	**++**	o	**++**
	Datenschema	+	**++**	o	**++**	o	**++**
	Metadaten	o	**++**	o	**++**	+	**++**
	Durchsuchbarkeit	+	**+**	o	**+**	++	**++**
	Betrachtung der Datenentstehung	o	**+**	++	**++**	o	**+**
	Behandlung orts-bezogener Daten	+	**+**	o	**o**	++	**++**
	Benutzer-schnittstelle	++	**++**	+	**+**	+	**+**
	Domänen-unabhängigkeit	o	**o**	++	**++**	+	**+**

Tabelle 6-1: Zusätzlicher Nutzen für die Leistungsmerkmale durch die Verwendung verwandter Arbeiten **mit** semantischen Annotationen

abgeleitet werden. Die so berechneten Fahrmanöver können ebenfalls im Semantic Database Browser visualisiert werden. Abbildung 6-1 zeigt das Zusammenspiel der verschiedenen Aspekte im Überblick. Dennoch sind semantische Datenbank-Annotationen ein generisches Werkzeug und lassen sich durch den Einsatz domänenspezifischer Ontologien für Anwendungsfälle aus unterschiedlichen Domänen einsetzen.

6.1.2 Bewertung

Tabelle 6-1 setzt die semantischen Annotationen und darauf aufbauende Konzepte in Bezug zu den *verwandten Arbeiten* aus den Bereichen *Informationssysteme für Fahrversuche*, *Scientific Workflow Systems* (SWS), *wissenschaftliche Datenhaltungssysteme* und *inhaltliche Indizierung und Annotation*. Die Tabelle orientiert sich an dem Vergleich dieser Arbeiten aus Abschnitt 2.2.6. Für die aus dem Umfeld des semantischen Web bekannte *inhaltliche Indizierung und Annotation* zur formalen Beschreibung von Medieninhalten gilt, dass sie für Web-, Text- und Multimedia-Inhalte üblich ist (vgl. Abschnitt 2.2.5). Die vorliegende Arbeit überträgt das Konzept der Annotation auf Messdaten, welche der zentrale Betrachtungsgegenstand der jeweils anderen Bereiche verwandter Arbeiten sind.

Aus Sicht der Architektur bilden mit semantischen Technologien beschriebene Messdaten eine von mehreren *Komponenten in einem verteilten Informationssystem* (Abschnitt 3.2.4). Der vorgestellte Ansatz ergänzt in einem verteilten System somit die Kategorien verwandter Arbeiten um verschiedene Aspekte und erlaubt ihre Methoden gleichzeitig auf einen gemeinsamen

Datenbestand anzuwenden. Damit deckt der kombinierte Einsatz der verschiedenen Werkzeugkategorien den gesamten Lebenszyklus der Daten von der Datenentstehung (SWS) über deren Analyse (Informationssysteme) bis zu deren Archivierung (Datenhaltungssysteme) ab. Für die gemeinsame Nutzung bilden die Metadaten in Form von semantischen Datenbank-Annotationen das Bindeglied, welche sowohl Informationen bezüglich der Datenentstehung als auch erweiterte Möglichkeiten für das Durchsuchen der archivierten Messdaten bereithalten.

Wie sich die Leistungsmerkmale der drei Kategorien durch die erzielten Ergebnisse verbessern, ist in Tab. 6-1 gegenübergestellt. Die *Datenverwaltung* und die *Interpretation der Dateninhalte* profitieren durch den Einsatz einer Datenbank für die Messdaten. Die formale Beschreibung der *Metadaten* mit semantischen Technologien wirkt sich positiv auf die Ausdrucksstärke des *Datenschemas* für Messdaten aus. Sowohl die Datenbank als auch die formalen Metadaten lassen sich sehr gut *durchsuchen*, was jedoch durch die einzelnen Benutzerschnittstellen auch unterstützt werden muss. Weiterhin lassen sich vorhandene Informationen zur *Datenentstehung* ideal mit den formalisierten Metadaten beschreiben.

6.2 Beantwortung der Forschungsfrage

> *Wie können in einer Datenbank gespeicherte numerische Daten (z.B. Zeitreihen) mit Domänenwissen, zur Wahrung der ursprünglichen und abgeleiteten Interpretation, skalierbar verknüpft werden und dem Anwender bedienfreundlich präsentiert werden, sodass diese Kontextinformationen in formaler Form bei der Arbeit mit den Daten verfügbar sind?*

Als Antwort auf die der Arbeit zugrunde liegenden Frage lässt sich festhalten, dass die semantischen Annotationen und die darauf aufbauenden und im Rahmen dieser Arbeit entwickelten Konzepte einen geeigneten Lösungsansatz darstellen. Die Annotationen stellen das Bindeglied zwischen den Messdaten und dem in formalen Modellen abgebildeten Domänenwissen dar. Für die formale Beschreibung werden Semantic Web Technologien und Ontologien eingesetzt, welche auf Prädikatenlogik basieren. Als Anwendungsbeispiel aus der Domäne von Fahrversuchen werden Fahrmanöver in aufgezeichneten Messfahrten formal beschrieben.

Die vorgestellte Lösung erreicht die geforderte Skalierbarkeit, indem die formalen Modelle aufgeteilt und nur die Teile betrachtet werden, welche gerade von Interesse sind. Der Anwender kann dank einer gemeinsamen Visualisierung von Messdaten und Annotationen komfortabel mit den semantischen Annotationen arbeiten. Die Visualisierung erfolgt hierbei anhand der Projektion der Annotationen auf die Messdaten. So kann der Anwender beispielsweise leicht die einzelnen im zeitlichen Verlauf aufgetretenen Fahrmanöver nachvollziehen.

Es folgt eine detailliertere Betrachtung in Bezug zu den aufgeschlüsselten Teilfragestellungen.

6.2.1 Konzept

Wie lassen sich Datenbankinhalte und semantische Technologien konzeptionell miteinander verknüpfen? Wie können klassische numerische Daten mit semantischen Annotationen in einem gemeinsamen Prozess behandelt werden?

Das konzeptionelle Bindeglied zwischen den Datenbankinhalten und den mit semantischen Technologien realisierten formalen Modellen stellen die semantischen Annotationen dar. Um ein Datenbank-Element in einem Modell referenzieren zu können, bilden URIs diese dynamisch je nach Bedarf ab, sodass semantische Technologien sie für formale Aussagen verwenden können. Auf dieser Basis sind beispielsweise Aussagen über aufgetretene Ereignisse in Fahrversuchen möglich. Diese Ereignisse bilden dann die Grundlage für die in einer Ontologie auf Basis von Automaten formalisierten Fahrmanöver.

Durch die geschickte Kombination einer Datenbank mit Semantic Web Technologien werden so die Vorteile zweier Technologien miteinander verbunden. Die Verwaltung sehr großer Datenmengen ist eine Stärke relationaler Datenbanken, aber allein mit Semantic Web Technologien bisher eher unbefriedigend. Dafür erlauben Ontologien eine Beschreibung komplexer Zusammenhänge in formalen Modellen. Die Referenzierung fremder Ressourcen (z.B. Webseiten), Integration verteilter Ontologie-Modelle (z.B. semantisches Wiki, DBpedia) und der Einsatz von Reasoning zur Beschreibung von Zeitreihen (z.B. Automaten) bieten weitere wertvolle Optionen. Beispielsweise ist die dynamische Ausführung von Automaten zur Berechnung von Fahrmanövern in Datenbanken allein mittels SQL und Views nicht möglich. Das erarbeitete Ergebnis ist somit mehr als nur die Summe zweier Technologien.

Methodisch werden semantische Annotationen parallel zur klassischen numerischen Datenverarbeitung in einem gemeinsamen Prozess eingesetzt und ergänzen diese sinnvoll um eine Wissensmodellierung. Ein kontinuierlicher Abgleich der Modellierungsebenen stellt deren Konsistenz sicher. Im Anwendungsfall der Fahrmanövermodellierung erfolgt ein stetiger Abgleich zwischen den Messdaten und den im Modell mittels Reasoning zu einer Messfahrt berechneten Fahrmanövern, indem diese miteinander in Beziehung gesetzt werden. Der Abgleich wird durch das umgesetzte Werkzeugkonzept Semantic Database Browser zur gemeinsamen Handhabung von Datenbanken mit semantischen Annotationen unterstützt. Die Annotationen erlauben eine ausdrucksstarke Dokumentation der Daten und erhöhen daher deren Qualität und langfristigen Wiederverwendungswert.

6.2.2 Architektur und Darstellung

Wie muss eine Architektur zur gemeinsamen Datenhaltung aussehen, damit sie bei großen Datenmengen skaliert? Wie kann die Einbettung der semantischen Zeitreihenbeschreibung zur gemeinsamen Benutzung in Anwendungen in einem verteilten System aussehen? Wie lässt sich eine semantische Beschreibung in Form von

Annotationen so visualisieren, dass die Datenbestände interaktiv exploriert werden können?

Aus Sicht der Architektur spielen die Verwaltung der Wissensmodelle, deren Speicherung und das Zusammenspiel der Funktionen in einem verteilten System eine wesentliche Rolle. Bezüglich der Wissensmodelle ist von Interesse, wo das eigentliche Wachstum der semantischen Annotationen stattfindet. Basierend auf der Erkenntnis, dass dieses Wachstum an die Experiment-Tabellen bzw. Versuchsfahrten gekoppelt ist, werden entsprechende Teilmodelle für einzelne Experimente gebildet. Diese erlauben die Modellkomplexität zu beherrschen, indem bei der Verwendung nur noch die relevanten Teilmodelle berücksichtigt werden. Dadurch ist die Komplexität unabhängig von einer ansteigenden Menge an Messdaten. Die Speicherarchitektur setzt die Anforderung um, diese Teilmodelle zu verwalten und parallel zu den Messdaten zu speichern.

Die Einbindung semantischer Annotationen in einem verteilten System kann in zwei verschiedene Richtungen erfolgen. Zum einen können fremde Ontologien in Bezug zur Datenbank gesetzt werden, alternativ werden die Datenbank-Elemente in anderen Ontologien referenziert. Der Zugriff auf die semantischen Annotationen kann auf drei unterschiedlichen Abstraktionsebenen stattfinden. Zum einen ist der direkte Zugriff über die speichernde Datenbank mittels SQL auf die Annotationen möglich. Zum anderen kann der Zugriff über das SIMDa-Framework in verschiedenen Applikationen erfolgen oder das auf dieser Basis realisierte Werkzeug Semantic Database Browser eingesetzt werden. Als dritte Möglichkeit erfolgt die Verknüpfung direkt auf logischer Ebene durch Referenzierung der Datenbank-Elemente in Ontologien. Somit ist für eine umfassende Wiederverwendbarkeit die Integration in diverse unterschiedliche Anwendungen möglich, die auf verschiedenen Abstraktionsebenen miteinander interagieren.

Die Annotationen lassen sich auf verschiedene Art durch das Konzept von (semantischen) Ebenen generisch visualisieren, welche eine interaktive Auswahl der betrachteten Annotationen und Modellinhalte erlauben. Die Visualisierung erfolgt durch die Projektion der so getroffenen Auswahl von Modellinhalten auf die Messdaten, deren Zeitachse oder einen Graphen. Insbesondere die Projektion auf die Zeitachse erlaubt dem Anwender, leicht den Verlauf aufgetretener Ereignisse oder Fahrmanöver nachzuvollziehen. Diese Anwendung einer Datenbank in Kombination mit semantischen Annotationen war bisher nicht in der Literatur sichtbar und eröffnet neue Anwendungsgebiete. Die erzielte Kombination ist somit mehr als lediglich die Summe zweier Komponenten.

6.2.3 Anwendung

Wie können Zeitreihen so mit semantischen Technologien beschrieben werden, dass die auftretenden Ereignisse über Reasoning miteinander in Beziehung gesetzt werden können? Wie eng ist bei den eingeführten Konzepten und Methoden die Kopplung an eine bestimmte Domäne?

Für die Beschreibung von Ereignissen in Zeitreihen können Automaten mit verschiedenen Erweiterungen verwendet werden, da diese ein leicht nachvollziehbares und leistungsfähiges Werkzeug bilden. Bei der Modellierung der Ereignisse ist darauf zu achten, dass eine möglichst kompakte Modellierung anzustreben ist, um eine möglichst aussagekräftige Darstellung zu erhalten, welche für das Reasoning vorteilhaft ist. Das Beispiel der Überführung von Fahrmanövern in eine auf Automaten basierende Ontologie demonstriert die direkte Umsetzbarkeit dieser Herangehensweise. Auf dieser Darstellung übernimmt das Reasoning die Berechnung des aktuellen Fahrmanövers anhand der während der Versuchsfahrt aufgetretenen Ereignisse. Weiterhin an dieser Ontologie und Versuchsfahrten durchgeführte Laufzeitmessungen zeigen, dass eine Anwendung in der Praxis sinnvoll möglich ist.

Ein Vorteil sämtlicher im Rahmen dieser Arbeit behandelten Konzepte ist die Unabhängigkeit von einer speziellen Domäne. Sie können daher in vielen anderen Bereichen zum Einsatz kommen. Hierfür ist lediglich eine Verwendung entsprechender domänenspezifischer Ontologien notwendig, wie dies beispielsweise für die Beschreibung von Fahrmanövern demonstriert wurde. Viele der behandelten Konzepte, wie die semantischen Annotationen und deren Visualisierung, beschränken sich nicht auf die alleinige Beschreibung von Zeitreihen oder Messdaten. Somit können diese auch beliebige andere Daten in einer Datenbank sinnvoll ergänzen.

6.3 Ausblick

In Zukunft werden verstärkt *Naturalistic Driving Studies* (NDS) durchgeführt werden, um Fahrer im regulären Straßenverkehr und unter natürlichen Bedingungen beurteilen zu können. Um für NDS relevante Aussagen treffen und interessante Situationen finden zu können, müssen viele Probanden Messdaten über lange Zeiträume liefern. Semantische Annotationen können in diesem Szenario einen wertvollen Beitrag leisten, um die Interpretationsfähigkeit dieser großen Menge von Messdaten langfristig zu gewährleisten und gewonnene Erkenntnisse zu sichern. Für diesen Anwendungsfall wird ebenfalls der Einsatz von Data-Mining-Verfahren zur Bewältigung der anfallenden Daten relevant werden.

Im Hinblick auf Zeitreihen bietet das im Abschnitt zu verwandten Arbeiten kurz eingeführte Data-Mining die Chance, weitgehend automatisiert markante Ereignisse zu detektieren und zu beschreiben. Diese Methode ergänzt sehr gut die Inhalte dieser Arbeit um weitergehende Aspekte, konnte aber aufgrund des Umfangs nicht vertieft werden. So können Ergebnisse des Data-Mining in Form von semantischen Annotationen ausgedrückt werden. Zusätzlich hinterlegte Regeln in Form von Ontologien können die erkannten Ergebnisse validieren. Diese Ergebnisse bilden dann in dieser Form Zwischenergebnisse, die für weitere Data-Mining-Schritte genutzt werden. Auch direkte Annotationen (z.B. zu Ausreißern) lassen sich so für das Data-Mining berücksichtigen. Damit können Annotationen eine Rolle als zusätzliche Datenreihen für das Data-Mining übernehmen.

Die Interpretationsfähigkeit der in NDS und anderen Experimenten anfallenden Daten lässt sich durch die Integration mit einem entsprechenden Wissensmanagement erhöhen. Hierbei werden in Ontologien Begriffssysteme formalisiert, welche in Zusammenhang mit den NDS und den dort zu untersuchenden Fahrsituationen stehen. Ein Wissensmanagement mit entsprechender Werkzeugunterstützung (z.B. semantisches Wiki, Ontologie-Werkzeuge) erlaubt es, eine menschenverständliche Beschreibung mit einer formalen Darstellung zu kombinieren. Diese kombinierte Darstellung fördert die wissenschaftliche Systematisierung der betrachteten Domäne. Der Einsatz der in dieser Arbeit vorgestellten Ansätze zur direkten Verknüpfung der betrachteten Daten mit einem Wissensmanagement ermöglicht es, eine möglichst hohe Integration zu erzielen. Durch diese Integration profitieren das Wissensmanagement und die Datenauswertung unmittelbar voneinander.

Die in dieser Arbeit vorgestellte Implementierung stellt eine erste Umsetzung dar, sodass weitere Ausarbeitungen möglich sind. In Zukunft wird das Werkzeug *Dominion-DataStore Control Center* weiter zur zentralen Benutzerschnittstelle für die Versuchsfahrtendatenbank Dominion-DataStore ausgebaut werden. Dieses Werkzeug wird den Zugriff auf die durchgeführten Fahrexperimente und NDS erlauben (vgl. Abschnitt 3.2.4). Das Dominion-DataStore Control Center bietet somit die Möglichkeit, die in dieser Arbeit vorgestellten Konzepte zu integrieren und in Zukunft weiter in der Praxis für Messdaten zu erproben und weiterzuentwickeln.

A Abkürzungsverzeichnis

API	Application Programming Interface
BExIS	Biodiversity-Exploratory Information System
BPEL	Business Process Execution Language
BSD	Berkeley Software Distribution
CAN	Controller Area Network
CSV	Comma-separated Values
CWM	Common Warehouse Metamodel
DFD	Deutsches Fernerkundungsdatenzentrum
DIMS	Data Information and Management System
DLR	Deutsches Zentrum für Luft- und Raumfahrt e. V.
DQM	Datenqualitätsmanagement
DWH	Data-Warehouse
EBNF	Erweiterte Backus-Naur-Form
FOAF	Friend of a Friend
FTP	File Transfer Protocol
GBIF	Global Biodiversity Information Facility
GIS	Geographisches Informationssystem
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
ILTER	International Long Term Ecological Research Network
ISO	International Organization for Standardization
ITS	Institute of Transportation Systems
JDBC	Java Database Connectivity
KB	Knowledge Base
LGPL	GNU Lesser General Public License
MORIS	Massachusetts Ocean Resource Information System
MPEG	Moving Picture Experts Group

MPL . . . . . Mozilla Public License
MUCOSA . . . . . A music content semantic annotator
N3 . . . . . Notation 3
NASA . . . . . National Aeronautics and Space Administration
NDS . . . . . Naturalistic Driving Studies
ODBC . . . . . Open Database Connectivity
OpenAFS . . . . . Open Andrew File System
ORM . . . . . Object-Relational Mapping
OSGi . . . . . Open Services Gateway initiative
OWL . . . . . Web Ontology Language
OWL DL . . . . . OWL Description Logic
PANGAEA . . . . . Data Publisher for Earth & Environmental Science
PDF . . . . . Portable Document Format
PL/SQL . . . . . Procedural Language/SQL
QUDT . . . . . Quantities, Units, Dimensions and Data Types in OWL and XML
RDF . . . . . Resource Description Framework
RDFS . . . . . RDF Schema
SDB . . . . . Semantic Database Browser
SIMDa . . . . . Semantic Integration of Measurement Data
SOA . . . . . Service Oriented Architecture
SPARQL . . . . . SPARQL Protocol and RDF Query Language
SQL . . . . . Structured Query Language
SSH . . . . . Secure Shell
SWRL . . . . . Semantic Web Rule Language
SWS . . . . . Scientific Workflow System
TS . . . . . Transportation Systems
TSM . . . . . Tivoli Storage Manager
UML . . . . . Unified Modeling Language
URI . . . . . Uniform Resource Identifier
URL . . . . . Uniform Resource Locator
URN . . . . . Uniform Resource Name
W3C . . . . . World Wide Web Consortium
WebDAV . . . . . Web-based Distributed Authoring and Versioning
WSDL . . . . . Web Services Description Language

WWW . . . . . World Wide Web

XHTML . . . . . Extensible Hypertext Markup Language

XLink . . . . . XML Linking Language

XMI . . . . . XML Metadata Interchange

XML . . . . . Extensible Markup Language

XPointer . . . . . XML Pointer Language

B Symbolverzeichnis

A . . . . . Automat

C . . . . . Menge von Klassen

c . . . . . Färbung einer Ebene

δ . . . . . Transitionsfunktion eines Automaten

E . . . . . Menge von Kanten

$\mathfrak{e}_{qq'}$ . . . . . Funktion eines Automaten zur Verarbeitung mehrerer Eingaben

f . . . . . Funktion einer Ebene zum Erstellen der Sicht

G . . . . . Graph

G_B . . . . . Basis-Graph-Modell

G_D . . . . . Datenbank-Graph-Modell

G_d . . . . . Abgeleiteter Graph einer Ebene

G_o . . . . . Originärer Graph einer Ebene

G_{RDF} . . . . . Graph-Modell mit externen RDF-Modellen

G_S . . . . . Graph-Modell mit Schema-Annotationen

G_U . . . . . Graph-Modell mit nutzerspezifischen Annotationen

I . . . . . Interpretation

i . . . . . Index einer Ebene

$\mathrm{I_{CEXT}}$ . . . . . Klassenextensionsfunktion

$\mathrm{I_{EXT}}$ . . . . . Extensionsfunktion

$\mathrm{I_L}$ . . . . . Abbildung auf Ressourcen

$\mathrm{I_S}$ . . . . . Abbildung auf Ressourcen und Properties

$\mathbb{L}$ . . . . . Menge der Ebenen

L . . . . . Ebene

LV . . . . . Menge von ungetypten Literalen

$\mathcal{L}$ . . . . . Menge von Literalen

M . . . . . Modell

o . . . . . Objekt eines Aussagetripels

P . . . . . Menge von Properties

p . . . . . Prädikat eines Aussagetripels

Q . . . . . Zustandsmenge eines Automaten

q_0 . . . . . Startzustand eines Automaten

R . . . . . Menge von Ressourcen

S . . . . . Menge von Statements

Σ . . . . . Alphabet eines Automaten

s . . . . . Subjekt eines Aussagetripels

σ . . . . . Buchstabe des Eingabealphabetes eines Automaten

V . . . . . Menge von Knoten

$\mathcal{V}$ . . . . . Vokabular

$\mathcal{V}_{RDF}$ . . . . . RDF-Vokabular

$\mathcal{V}_{RDFS}$ . . . . . RDFS-Vokabular

C Literaturverzeichnis

[ABI11] AUER, SÖREN, CHRISTIAN BIZER und KINGSLEY IDEHEN: *DBpedia*. WWW: `http://dbpedia.org`, Januar 2011. (03.01.2011).

[ABJ+04] ALTINTAS, ILKAY, CHAD BERKLEY, EFRAT JAEGER, MATTHEW JONES, BERTRAM LUDÄSCHER und STEVE MOCK: *Kepler: An Extensible System for Design and Execution of Scientific Workflows*. In: *International Conference on Scientific and Statistical Database Management (SSDBM)*, Seiten 423 – 424, Juni 2004.

[ABK+07] AUER, SÖREN, CHRISTIAN BIZER, GEORGI KOBILAROV, JENS LEHMANN, RICHARD CYGANIAK und ZACHARY IVES: *DBpedia: A Nucleus for a Web of Open Data*. In: *The Semantic Web — 6th International Semantic Web Conference*, Band 4825 der Reihe *Lecture Notes in Computer Science*, Seiten 722 – 735, November 2007.

[ACKP01] ALEXAKI, SOFIA, VASSILIS CHRISTOPHIDES, GREG KARVOUNARAKIS und DIMITRIS PLEXOUSAKIS: *On Storing Voluminous RDF Descriptions: The Case of Web Portal Catalogs*. In: *International Workshop on the Web and Databases*, Seiten 24 – 25, 2001.

[ADD10] AUER, SÖREN, RAPHAEL DOEHRING und SEBASTIAN DIETZOLD: *LESS — Template-Based Syndication and Presentation of Linked Data*. The Semantic Web: Research and Applications, 6089/2010:211 – 224, 2010.

[AG08] ANGLES, RENZO und CLAUDIO GUTIERREZ: *Survey of graph database models*. ACM Computing Surveys (CSUR), 40(1):1 – 39, Februar 2008.

[AGP10] ARENAS, MARCELO, CLAUDIO GUTIERREZ und JORGE PEREZ: *On the Semantics of SPARQL*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 13, Seiten 281 – 307. Springer, 2010.

[AL99] ALAVI, MARYAM und DOROTHY LEIDNER: *Knowledge Management Systems: Issues, Challenges, and Benefits*. Communications of the Association for Information Systems, 1:2 – 37, 1999.

[Alf14a] ALFRED-WEGENER-INSTITUT, HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG: *Data Publisher for Earth & Environmental Science*. WWW: `http://www.pangaea.de`, Januar 2014. (15.01.2014).

[Alf14b] ALFRED-WEGENER-INSTITUT, HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG: *PangaWiki*. WWW: `http://wiki.pangaea.de`, Januar 2014. (15.01.2014).

[All83] ALLEN, JAMES F.: *Maintaining knowledge about temporal intervals*. Communications of the ACM, 26(11):832 – 843, November 1983.

[And86] ANDREWS, PETER BRUCE: *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[ANP10] ATZENI, PAOLO, PIERLUIGI DEL NOSTRO und STEFANO PAOLOZZI: *MIDST: Interoperability for Semantic Annotations*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 20, Seiten 479 – 500. Springer, 2010.

[AS08] ADUNA-SOFTWARE: *OpenRDF.org — ... home of Sesame*. WWW: `http://www.openrdf.org/`, Oktober 2008. (10.10.2008).

[Asa08] ASANGANA, KILIAN: *Metadata Annotation of Intensions and Extensions of Different Data Models*. Diplomarbeit, Carl von Ossietzky Universität Oldenburg, Juni 2008.

[AvMH02] AUTH, GUNNAR, EITEL VON MAUR und MARKUS HELFERT: *A Model-based Software Architecture for Metadata Management in Data Warehouse Systems*. In: *5th International Conference on Business Information Systems (BIS '02)*, 2002.

[BB08] BECKETT, DAVE und JEEN BROEKSTRA: *SPARQL Query Results XML Format*. W3C Recommendation: `http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/`, Januar 2008. (25.04.2010).

[BCH+08] BAUMANN, M., H. COLONIUS, H. HUNGAR, F. KÖSTER, M. LANGNER, A. LÜDTKE, C. MÖBIUS, J. PEINKE, S. PUCH, C. SCHIESSL, R. STEENKEN und L. WEBER: *Integrated Modelling for Safe Transportation — Driver modeling and driver experiments*. In: *2te Fachtagung Fahrermodellierung*, 2008.

[Bec04] BECKETT, DAVE: *RDF/XML Syntax Specification (Revised)*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`, Februar 2004. (01.05.2010).

[BETT98] BATTISTA, GIUSEPPE DI, PETER EADES, ROBERTO TAMASSIA und IOANNIS G. TOLLIS: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.

[BG03] BECKETT, DAVE und JAN GRANTT: *SWAD-Europe Deliverable 10.2: Mapping Semantic Web Data with RDBMSes*. Technischer Bericht, W3C, Januar 2003.

[BG04] BRICKLEY, DAN und R.V. GUHA: *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`, Februar 2004. (03.01.2011).

[BG09] BAUER, ANDREAS und HOLGER GÜNZEL: *Data-Warehouse-Systeme. Architektur, Entwicklung, Anwendung*. Dpunkt Verlag, 3. Auflage, 2009.

[BHJV08] BOCK, JÜRGEN, PETER HAASE, QIU JI und RAPHAEL VOLZ: *Benchmarking OWL Reasoners*. In: *Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*, 2008.

[BHL+09] BRAY, TIM, DAVE HOLLANDER, ANDREW LAYMAN, RICHARD TOBIN und HENRY S. THOMPSON: *Namespaces in XML 1.0 (Third Edition)*. W3C Recommendation: `http://www.w3.org/TR/2009/REC-xml-names-20091208/`, Dezember 2009. (30.03.2010).

[BHM+12] BOOTH, DAVID, HUGO HAAS, FRANCIS MCCABE, ERIC NEWCOMER, MICHAEL CHAMPION, CHRIS FERRIS und DAVID ORCHARD: *Web Services Architecture*. W3C Working Group Note: `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/`, Februar 2012. (21.06.2010).

[BHMS09] BECKER, UWE, MATTHIAS HÜBNER, HANSJÖRG MANZ und ECKEHARD SCHNIEDER: *Modellierung domänenübergreifender Terminologien für satellitengestützte Positionsbestimmungssysteme durch ontologiebasierte Systementwicklung*. In: *POSNAV 2009*, Oktober 2009.

[BHS09] BAADER, FRANZ, IAN HORROCKS und ULRIKE SATTLER: *Description Logics*. In: *Handbook on Ontologies*, Kapitel 1, Seiten 21 – 43. Springer, 2009.

[Biz03] BIZER, CHRISTIAN: *D2R MAP — A Database to RDF Mapping Language*. In: *WWW2003*, Mai 2003.

[BKT01] BUNEMAN, PETER, SANJEEV KHANNA und WANG-CHIEW TAN: *Why and Where: A Characterization of Data Provenance*. Lecture Notes in Computer Science, 1973:316 – 330, 2001.

[BKvH02] BROEKSTRA, JEEN, ARJOHN KAMPMAN und FRANK VAN HARMELEN: *Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema*. In: *First International Semantic Web Conference*, Band 2342 der Reihe *Lecture Notes in Computer Science*, Seiten 54 – 68, Juni 2002.

[BL98] BERNERS-LEE, TIM: *Relational Databases on the Semantic Web*. WWW: `http://www.w3.org/DesignIssues/RDB-RDF.html`, September 1998. (10.07.2010).

[BL06a] BERNERS-LEE, TIM: *Linked Data*. WWW: `http://www.w3.org/DesignIssues/LinkedData.html`, Juli 2006. (10.07.2010).

[BL06b] BERNERS-LEE, TIM: *Notation 3 — An readable language for data on the Web*. WWW: `http://www.w3.org/DesignIssues/Notation3.html`, März 2006. (08.04.2009).

[BL07] BERNERS-LEE, TIM: *Giant Global Graph*. WWW: `http://dig.csail.mit.edu/breadcrumbs/node/215`, November 2007. (20.12.2010).

[BL09] BP LOGIX, INC: *iMarkup Client*. WWW: `http://www.bplogix.com/client/`, November 2009. (24.11.2009).

[BLFM05] BERNERS-LEE, TIM, R. FIELDING und L. MASINTER: *Uniform Resource Identifier (URI): Generic Syntax — IETF RFC 3986*. WWW: `http://tools.ietf.org/html/rfc3986`, Januar 2005. (31.03.2010).

[BLHL01] BERNERS-LEE, TIM, JAMES HENDLER und ORA LASSILA: *The Semantic Web*. Scientific American, 284(5):34 – 43, Mai 2001.

[BOCGP04] BARRASA, JESÚS, ÓSCAR CORCHO und ASUNCIÓN GÓMEZ-PÉREZ: *R2O, an Extensible and Semantically Based Database-to-ontology Mapping Language*. In: *Second Workshop on Semantic Web and Databases (SWDB2004)*, August 2004.

[Bos02] BOSE, RAJENDRA: *A Conceptual Framework for Composing and Managing Scientific Data Lineage*. In: *International Conference on Scientific and Statistical Database Management*, Band 14, Seiten 15 – 19, November 2002.

[BP09] BLUMAUER, ANDREAS und TASSILO PELLEGRINI: *Semantic Web Revisited - Eine kurze Einführung in das Social Semantic Web*. In: *Social Semantic Web*, Seiten 3 – 22. Springer Berlin Heidelberg, 2009.

[BPM04] BIRON, PAUL V., KAISER PERMANENTE und ASHOK MALHOTRA: *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`, Oktober 2004. (29.03.2010).

[BPS+05] BLOEHDORN, STEPHAN, KOSMAS PETRIDIS, CARSTEN SAATHOFF, NIKOS SIMOU, VASSILIS TZOUVARAS, YANNIS AVRITHIS, SIEGFRIED HANDSCHUH, YIANNIS KOMPATSIARIS, STEFFEN STAAB und MICHAEL G. STRINTZIS: *Semantic Annotation of Images and Videos for Multimedia Analysis*. In: *Second European Semantic Web Conference, ESWC 2005*, Seiten 592 – 607, Mai 2005.

[BPSM+06] BRAY, TIM, JEAN PAOLI, C. M. SPERBERG-MCQUEEN, EVE MALER, FRANÇOIS YERGEAU und JOHN COWAN: *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C Recommendation: `http://www.w3.org/TR/2006/REC-xml11-20060816/`, August 2006. (30.03.2010).

[Bri10] BRICKLEY, DAN: *The Friend of a Friend (FOAF) project*. WWW: `http://www.foaf-project.org/`, April 2010. (03.04.2010).

[BRMR01] BÖTTCHER, MARTIN, RALF REISSIG, EBERHARD MIKUSCH und CHRISTOPH RECK: *Processing Management Tools for Earth Observation Products at DLR-DFD*. In: ESTEC, EUROPEAN SPACE AGENCY PUBLICATIONS DIVISION (Herausgeber): *Data Systems in Aerospace, Nice, 2001*, Band SP-483. European Space Agency, 2001.

[BS98] BARRY, DOUGLAS und TORSTEN STANIENDA: *Solving the Java Object Storage Problem*. Computer, 31:33 – 40, November 1998.

[BTX+09] BOCK, JÜRGEN, TUVSHINTUR TSERENDORJ, YONGCHUN XU, JENS WISSMANN und STEPHAN GRIMM: *A Reasoning Broker Framework for Protégé*. In: *11th International Protégé Conference*, Seiten 48 – 52, 2009.

[Bun07] BUNZEL, BENJAMIN: *Konzeption und Realisierung eines GIS-Frameworks zur flexiblen Darstellung ortsbezogener Messdaten*. Diplomarbeit, Fachhochschule Braunschweig/Wolfenbüttel, August 2007.

[BvHH+04] BECHHOFER, SEAN, FRANK VAN HARMELEN, JIM HENDLER, IAN HORROCKS, DEBORAH L. MCGUINNESS, PETER F. PATEL-SCHNEIDER und LYNN ANDREA STEIN: *OWL Web Ontology Language Reference*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, Februar 2004. (01.05.2010).

[BW00] BAER, DIETER und MATTHIAS WERMKE: *Duden - Das große Fremdwörterbuch*. Bibliographisches Institut, 2. Auflage, 2000.

[BWM11] BELL, COLIN, GERD WAGNER und ROB MANNING: *SQuirreL SQL: Universal SQL Client*. WWW: `http://squirrel-sql.sourceforge.net/`, Januar 2011. (31.01.2011).

[Car09] CARDIFF UNIVERSITY: *Triana — The Open Source Problem Solving Environment*. WWW: `http://www.trianacode.org/`, November 2009. (07.11.2009).

[CB74] CHAMBERLIN, DONALD D. und RAYMOND F. BOYCE: *SEQUEL: A Structured English Query Language*. In: *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control*, Mai 1974.

[CCD+04] CALIMLIM, MANUEL, JIM CORDES, ALAN DEMERS, JULIA DENEVA, JOHANNES GEHRKE, DAN KIFER, MIREK RIEDEWALD und JAYAVEL SHANMUGASUNDARAM: *A Vision for PetaByte Data Management and Analysis Services for the Arecibo Telescope*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 27(4):12 – 20, 2004.

[CCK+00] CHAPMAN, PETE, JULIAN CLINTON, RANDY KERBER, THOMAS KHABAZA, THOMAS REINARTZ, COLIN SHEARER und RÜDIGER WIRTH: *CRISP-DM 1.0: Step-by-step data mining guide*, 2000.

[CD97] CHAUDHURI, SURAJIT und UMESHWAR DAYAL: *An overview of data warehousing and OLAP technology*. SIGMOD Rec., 26(1):65–74, 1997.

[CDD+04] CARROLL, JEREMY J., IAN DICKINSON, CHRIS DOLLIN, DAVE REYNOLDS, ANDY SEABORNE und KEVIN WILKINSON: *Jena: Implementing the Semantic Web Recommendations*. In: *13th International World Wide Web Conference*, Mai 2004.

[Cer03] CERUZZI, PAUL E.: *A History of Modern Computing*. MIT Press, 2. Auflage, Mai 2003.

[CFT08] CLARK, KENDALL GRANT, LEE FEIGENBAUM und ELIAS TORRES: *SPARQL Protocol for RDF*. W3C Recommendation: `http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/`, Januar 2008. (25.04.2010).

[CG07] CLAUDIO GUTIERREZ, CARLOS A. HURTADO, ALEJANDRO A. VAISMAN: *Introducing Time into RDF*. In: *IEEE Transactions on Knowledge and Data Engineering*,

Band 19, Seiten 207 – 218, Februar 2007.

[CGL+99] CALVANESE, DIEGO, GIUSEPPE DE GIACOMO, MAURIZIO LENZERINI, DANIELE NARDI und RICCARDO ROSATI: *A Principled Approach to Data Integration and Reconciliation in DataWarehousing*. In: *International Workshop on Design and Management of Data Warehouse (DMDW'99)*, Juni 1999.

[CJ07] CHAPMAN, ADRIANE und H.V. JAGADISH: *Issues in Building Practical Provenance Systems*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 30(4):38 – 34, 2007.

[Cla10] CLARK & PARSIA, LLC: *OwlSight: A very lightweight OWL ontology browser*. WWW: `http://pellet.owldl.com/ontology-browser/`, Mai 2010. (05.05.2010).

[CLR01] CORMEN, THOMAS H., CHARLES E. LEISERSON und RONALD L. RIVEST: *Introduction into Algorithms*. MIT Press, 2. Auflage, 2001.

[Coa10] COASTAL ZONE MANAGEMENT: *Massachusetts Ocean Resource Information System (MORIS)*. WWW: `http://www.mass.gov/czm/mapping/index.htm`, Dezember 2010. (18.12.2010).

[CP06] CARROLL, JEREMY J. und JEFF Z. PAN: *XML Schema Datatypes in RDF and OWL*. W3C Working Group Note: `http://www.w3.org/TR/2006/NOTE-swbp-xsch-datatypes-20060314/`, März 2006. (29.03.2010).

[CR92] COLMERAUER, ALAIN und PHILIPPE ROUSSEL: *The birth of Prolog*. In: *History of programming languages — II*, Seiten 331 – 367, November 1992.

[CRPC09] CONSTANTINOU, CHARALAMBOS, SYMEON RETALIS, GEORGE PAPADOPOULOS und VRASIDAS CHARALAMBOS: *Combining Streaming Media and Collaborative Elements to Support Lifelong Learning*. In: *Intelligent Collaborative e-Learning Systems and Applications*, Band 246 der Reihe *Studies in Computational Intelligence*, Seiten 19 – 36. Springer, 2009.

[CW00] CUI, YINGWEI und JENNIFER WIDOM: *Practical Lineage Tracing in Data Warehouses*. In: *16th International Conference on Data Engineering (ICDE'00)*, Seite 367 ff., 2000.

[CW01] CUI, YINGWEI und JENNIFER WIDOM: *Lineage Tracing for General Data Warehouse Transformations*. In: *Proceedings of the 27th VLDB Conference*, 2001.

[CZ00] CHEN, CINDY XINMIN und CARLO ZANIOLO: *SQLST: A Spatio-Temporal Data Model and Query Language*. In: *Conceptual Modeling — ER 2000*, Band 1920 der Reihe *Lecture Notes in Computer Science*, Seiten 96 – 111. Springer, Oktober 2000.

[DBE+07] DAVIDSON, SUSAN, SARAH COHEN BOULAKIA, ANAT EYAL, BERTRAM LUDÄSCHER, TIMOTHY MCPHILLIPS, SHAWN BOWERS, MANISH KUMAR ANAND und JULIANA FREIRE: *Provenance in Scientific Workflow Systems*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 30(4):44 – 50, Dezember 2007.

[DGR+02] DIEPENBROEK, MICHAEL, HANNES GROBE, MANFRED REINKE, UWE SCHINDLER, REINER SCHLITZER, RAINER SIEGER und GEROLD WEFER: *PANGAEA — an information system for environmental sciences*. Computers & Geosciences, 28(10):1201 – 1210, Dezember 2002.

[DLNS94] DONINI, FRANCESCO M., MAURIZIO LENZERINI, DANIELE NARDI und ANDREA SCHAERF: *Deduction in Concept Languages: From Subsumption to Instance*

Checking. Journal of Logic and Computation, 4:423 – 452, 1994.

[DMD01] DeRose, Steve, Eve Maler und Ron Daniel: *XML Pointer Language (XPointer) Version 1.0*. W3C Last Call Working Draft: `http://www.w3.org/TR/2001/WD-xptr-20010108/`, Januar 2001. (21.06.2010).

[DMO01] DeRose, Steve, Eve Maler und David Orchard: *XML Linking Language (XLink) Version 1.0*. W3C Recommendation: `http://www.w3.org/TR/2001/REC-xlink-20010627/`, Juni 2001. (21.06.2010).

[Dub09] Dublin Core Metadata Initiative: *Information and documentation — The Dublin Core metadata element set (ISO 15836:2009)*. Technischer Bericht, International Organization of Standardization (ISO), 2009.

[DZFJ05] Ding, Li, Lina Zhou, Tim Finin und Anupam Joshi: *How the Semantic Web is Being Used: An Analysis of FOAF Documents*. In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Januar 2005.

[EA12] Esling, Philippe und Carlos Agon: *Time-series data mining*. ACM Computing Surveys (CSUR), 45(1), November 2012.

[Eck11] Eckstein, Silke: *Informationsmanagement in der Systembiologie: Datenbanken, Integration, Modellierung*. Springer, 2011.

[Edw04] Edwards, James L.: *Research and Societal Benefits of the Global Biodiversity Information Facility*. BioScience, 54(6):485 – 486, Juni 2004.

[EEM+07] Ehrich, Hans-Dieter, Silke Eckstein, Brigitte Mathiak, Andreas Kupfer und Claudia Täubner: *Bioinformatik: Erkenntnisse aus der Datenflut?* In: *Abhandlungen der Braunschweigischen Wissenschaftlichen Gesellschaft*, Band LVII, Seiten 9 – 34. J. Cramer Verlag, 2007.

[Ehr07] Ehrich, Frank: *Einsatz von räumlichen Datenbanken in der Verkehrs- und Unfallforschung*. In: *Deutsche Oracle Anwendergruppe (DOAG) Special Interest Group (SIG) Spatial*, Frankfurt, Februar 2007.

[Eif10] Eifrem, Emil: *Neo4j — the benefits of graph databases*. WWW: `http://www.slideshare.net/directi/neo4j-and-the-benefits-of-graph-dbs-3-3325734`, März 2010. (20.12.2010).

[EM10] Erling, Orri und Ivan Mikhailov: *Virtuoso: RDF Support in Native RDBMS*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 21, Seiten 501 – 519. Springer, 2010.

[ES02] Erwig, Martin und Markus Schneider: *STQL — A Spatio-Temporal Query Language*, Band 699 der Reihe *The Springer International Series in Engineering and Computer Science*, Kapitel 6, Seiten 105 – 126. Springer, 2002.

[ESL+05a] Ervin, R., J. Sayer, D. LeBlanc, S. Bogard, M. Mefford, M. Hagan, Z. Bareket und C. Winkler: *Automotive Collision Avoidance System Field Operational Test — Methodology and Results Appendices*. Technischer Bericht, University of Michigan, Transportation Research Institute and General Motors, Research and Development Center, The University of Michigan; Transportation Research Institute; 2901 Baxter Road, Ann Arbor, MI 48109-2150, August 2005.

[ESL+05b] Ervin, R., J. Sayer, D. LeBlanc, S. Bogard, M. Mefford, M. Hagan, Z. Bareket und C. Winkler: *Automotive Collision Avoidance System Field Operational Test — Report: Methodology and Results*. Technischer Bericht, University of Michigan, Transportation Research Institute and General Motors, Research

and Development Center, The University of Michigan; Transportation Research Institute; 2901 Baxter Road, Ann Arbor, MI 48109-2150, August 2005.

[ET06] EHRICH, FRANK und AXEL TENZER: *Speicherung von zeitkontinuierlichen Sensordaten (Laserscanner) am Beispiel der Fahrerleistungsdatenbank*. In: *5. Oldenburger 3D-Tage: Optische Messtechnik-Photogrammetrie-Laserscanning*, 2006.

[Eur06] EUROPÄISCHES KOMITEE FÜR NORMUNG: *DIN EN ISO 9241: Ergonomie der Mensch-System-Interaktion*. ISO Norm, 2006.

[Fel98] FELLBAUM, CHRISTIANE (Herausgeber): *Wordnet: An Electronic Lexical Database*. MIT Press, 1998.

[FES+98] FANCHER, P., R. ERVIN, J. SAYER, M. HAGAN, S. BOGARD, Z. BAREKET, M. MEFFORD und J. HAUGEN: *Intelligent Cruise Control Field Operational Test (Final Report)*. Technischer Bericht, The University of Michigan, Transportation Research Institute, 2901 Baxter Road, Ann Arbor, MI 48109-2150, Mai 1998.

[FHH04] FIKES, RICHARD, PAT HAYES und IAN HORROCKS: *OWL-QL: A Language for Deductive Query Answering on the Semantic Web*. Journal of Web Semantics, 2:19 – 29, Dezember 2004.

[FMK10] FRASINCAR, FLAVIUS, VIOREL MILEA und UZAY KAYMAK: *tOWL: Integrating Time in OWL*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 11, Seiten 225 – 246. Springer, 2010.

[FPSS96] FAYYAD, USAMA, GREGORY PIATETSKY-SHAPIRO und PADHRAIC SMYTH: *From Data Mining to Knowledge Discovery in Databases*. In: *AI Magazine*, Seiten 37 – 54, 1996.

[Fre99] FREE SOFTWARE FOUNDATION: *GNU Lesser General Public License, Version 2.1*. WWW: `http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html`, Februar 1999. (22.03.2014).

[FU10] FISCHER, FLORIAN und GULAY UNEL: *Reasoning in Semantic Web-based Systems*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 7, Seiten 127 – 146. Springer, 2010.

[GA99] GILLMAN, DANIEL W. und MARTIN V. APPEL: *Statistical Metadata Research at the Census Bureau*. In: *FCSM Conference*, 1999.

[GAWPL96] GILLMAN, DANIEL W., MARTIN V. APPEL und JR. WILLIAM P. LAPLANT: *Statistical Metadata Management: A Standards Based Approach*. In: *Proceedings of the Survey Research Methods Section, American Statistical Association*, Seite 55 ff., 1996.

[Gei99] GEIGER, KYLE: *Inside ODBC*. Microsoft Press, 1999.

[GG95] GUARINO, NICOLA und PIERDANIELE GIARETTA: *Ontologies and Knowledge Bases - Towards a Terminological Clarification*, Kapitel 2, Seiten 25 – 32. IOS Press, 1995.

[GHH08] GAČNIK, JAN, OLIVER HÄGER und MARCO HANNIBAL: *A Service-Oriented System Architecture For The Human Centered Design Of Intelligent Transportation Systems*. In: *Human Centered Design, Service-Oriented Architecture, Advanced Driver Assistance Systems*, Lyon, Frankreich, April 2008.

[GHHK08] GAČNIK, JAN, OLIVER HÄGER, MARCO HANNIBAL und FRANK KÖSTER: *Service-Oriented Architecture For Future Driver Assistance Systems*. In: *FISITA 2008 Automotive World Congress*, September 2008.

[GHJV04] GAMMA, ERICH, RICHARD HELM, RALPH JOHNSON und JOHN VLISSIDES: *Entwurfsmuster*. Addison Wesley Pub Co Inc, 2004.

[GHM+09] GRAU, BERNARDO CUENCA, IAN HORROCKS, BORIS MOTIK, BIJAN PARSIA, PETER PATEL-SCHNEIDER und ULRIKE SATTLER: *OWL 2: The next step for OWL*. Journal of Web Semantics, 6(4):309 – 322, November 2009.

[GHT06] GARDINER, TOM, IAN HORROCKS und DMITRY TSARKOV: *Automated Benchmarking of Description Logic Reasoners*. In: *Description Logics Workshop*, 2006.

[GJBK08] GAČNIK, JAN, HENNING JOST, DANIEL BEISEL und FRANK KÖSTER: *DESCAS — Design Process for the Development of Safety-Critical Advanced Driver Assistance Systems*. In: *FORMS 2008*, 2008.

[GM78] GALLAIRE, HERVE und JACK MINKER (Herausgeber): *Logic and Data Bases*. Plenum Press, New York, 1978.

[GM05] GRIMM, STEPHAN und BORIS MOTIK: *Closed World Reasoning in the Semantic Web through Epistemic Operators*. In: *Workshop on OWL: Experiences and Directions*, Band 188, November 2005.

[GMF+03] GENNARI, JOHN H., MARK A. MUSEN, RAY W. FERGERSON, WILLIAM E. GROSSO, MONICA CRUBÉZY, HENRIK ERIKSSON, NATALYA F. NOY und SAMSON W. TU: *The evolution of Protégé: an environment for knowledge-based systems development*. International Journal of Human-Computer Studies, 58(1):89 – 123, Januar 2003.

[GMN84] GALLAIRE, HERVE, JACK MINKER und JEAN-MARIE NICOLAS: *Logic and Databases: A Deductive Approach*. In: *ACM Computing Surveys*, Band 16, Seiten 153 – 185, Juni 1984.

[Gor06] GORKE, BASTIAN: *XML-Datenbanken in der Praxis*. Bomots-Verlag, 1. Auflage, 2006.

[GP09] GANGEMI, ALDO und VALENTINA PRESUTTI: *Ontology Design Patterns*. In: *Handbook on Ontologies*, Kapitel 11, Seiten 221 – 243. Springer, 2009.

[GS01] GERTZ, MICHAEL und KAI-UWE SATTLER: *A Model and Architecture for Conceptualized Data Annotations*. Technischer Bericht CSE-2001-11, Department of Computer Science, University of California, Davis, November 2001.

[GS02a] GERTZ, MICHAEL und KAI-UWE SATTLER: *Integrating Scientific Data through External, Concept-based Annotations*. In: *Second International Workshop Data Integration over the Web*, Seiten 87 – 102, 2002.

[GS02b] GRAND, BENEDICTE LE und MICHEL SOTO: *Visualisation of the Semantic Web: Topic Maps visualisation*. In: *Information Visualisation*, Seiten 344 – 349, November 2002.

[Hay04] HAYES, PETER: *RDF Semantics*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, Februar 2004. (04.03.2010).

[HB09] HORRIDGE, MATTHEW und SEAN BECHHOFER: *The OWL API: A Java API for Working with OWL 2 Ontologies*. In: *OWLED 2009, 6th OWL Experienced and Directions Workshop*, Juni 2009.

[HBN07] HORRIDGE, MATTHEW, SEAN BECHHOFER und OLAF NOPPENS: *Igniting the OWL 1.1 Touch Paper: The OWL API*. In: *OWLED 2007, 3rd OWL Experienced and Directions Workshop*, Juni 2007.

[Heu97] HEUER, ANDREAS: *Objektorientierte Datenbanken - Konzepte, Modelle, Stan-*

dards und Systeme. Addison-Wesley, 2. Auflage, 1997.

[HHMW12] HAARSLEV, VOLKER, KAY HIDDE, RALF MÖLLER und MICHAEL WESSEL: *The RacerPro knowledge representation and reasoning system*. Semantic Web, 3(3):267 – 277, 2012.

[HISW10] HALPIN, HARRY, RENATO IANNELLA, BRIAN SUDA und NORMAN WALSH: *Representing vCard Objects in RDF*. W3C Member Submission: `http://www.w3.org/Submission/2010/SUBM-vcard-rdf-20100120/`, Januar 2010. (03.04.2010).

[HKR08] HITZLER, PASCAL, MARKUS KRÖTZSCH und SEBASTIAN RUDOLPH: *Semantic Web — Grundlagen*. eXamen.press. Springer-Verlag, 1. Auflage, 2008.

[HLS+08] HAASE, PETER, HOLGER LEWEN, RUDI STUDER, DUC THANH TRAN, MICHAEL ERDMANN, MATHIEU D'AQUIN und ENRICO MOTTA: *The NeOn Ontology Engineering Toolkit*. In: *17th International World Wide Web Conference*, April 2008.

[HM03] HAARSLEV, VOLKER und RALF MÖLLER: *Racer: An OWL Reasoning Agent for the Semantic Web*. In: *Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with the 2003 IEEE/WIC International Conference on Web Intelligence*, Seiten 27 – 36, Oktober 2003.

[HMP+06] HOVY, EDUARD, MITCHELL MARCUS, MARTHA PALMER, LANCE RAMSHAW und RALPH WEISCHEDEL: *OntoNotes: The 90% Solution*. In: *Human Language Technology Conference*, Seiten 57 – 60, 2006.

[HNKR10] HEIMANN, DENNIS, JENS NIESCHULZE und BIRGITTA KÖNIG-RIES: *A flexible statistics web processing service–added value for information systems for experiment data*. Integrative Bioinformatics, 7(1):1 – 15, April 2010.

[HOCM+05] HERRERA, PERFECTO, ÒSCAR CELMA, JORDI MASSAGUER, PEDRO CANO, EMILIA GÓMEZ, FABIEN GOUYON, MARKUS KOPPENBERGER, DAVID GARCÍA, JOSÉ-PEDRO GARCÍA und NICOLAS WACK: *MUCOSA: A Music Content Semantic Annotator*. In: *International Conference on Music Information Retrieval 2005*, September 2005.

[HP06] HOBBS, JERRY R. und FENG PAN: *Time Ontology in OWL*. W3C Working Draft: `http://www.w3.org/TR/2006/WD-owl-time-20060927/`, September 2006. (02.01.2010).

[HP09] HITZLER, PASCAL und BIJAN PARSIA: *Ontologies and Rules*. In: STAAB, STEFFEN und RUDI STUDER (Herausgeber): *Handbook on Ontologies*, Kapitel 6, Seiten 111 – 132. Springer, 2009.

[HPPSH05] HORROCKS, IAN, BIJAN PARSIA, PETER PATEL-SCHNEIDER und JAMES HENDLER: *Semantic Web Architecture: Stack or Two Towers?* In: *Proc. of Principles and Practice of Semantic Web Reasoning*, Seiten 37 – 41, 2005.

[HPS09] HORRIDGE, MATTHEW und PETER F. PATEL-SCHNEIDER: *OWL 2 Web Ontology Language — Manchester Syntax*. W3C Working Group Note: `http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/`, Oktober 2009. (08.04.2010).

[HPSB+04] HORROCKS, IAN, PETER F. PATEL-SCHNEIDER, HAROLD BOLEY, SAID TABET, BENJAMIN GROSOF und MIKE DEAN: *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission: `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`, Mai 2004. (12.02.2011).

[HPSvH03] HORROCKS, IAN, PETER F. PATEL-SCHNEIDER und FRANK VAN HARMELEN: *From*

$\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language. Journal of Web Semantics, 1(1):7–26, 2003.

[HSWW03] HOLLINK, LAURA, GUUS SCHREIBER, JAN WIELEMAKER und BOB WIELINGA: *Semantic Annotation of Image Collections*. In: *Workshop on Knowledge Capture and Semantic Annotation (KCAP)*, 2003.

[HTL07] HUMMEL, BRITTA, WERNER THIEMANN und IRINA LULCHEVA: *Description Logic for Vision-Based Intersection Understanding*. In: *Cognitive Systems with Interactive Sensors (COGIS)*, 2007.

[HW08] HARTMANN, STEFAN und MARTIN WEBER: *Semantisch homogene Beschreibung von Data-Warehouse-Metadaten mit RDF*. In: *Multikonferenz Wirtschaftsinformatik*, 2008.

[HWAK08] HUANG, DAYONG, XINQI WANG, GABRIELLE ALLEN und TEVFIK KOSAR: *Semantic Enabled Metadata Framework for Data Grids*. In: *International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC-2008)*, März 2008.

[HWTS08] HARRISON, ANDREW, IAN WANG, IAN TAYLOR und MATTHEW SHIELDS: *WS-RF Workflow in Triana*. International Journal of High Performance Computing Applications, 22(3):268 – 283, 2008.

[HYD07] HUMMEL, BRITTA, ZONGRU YANG und CHRISTIAN DUCHOW: *Kreuzungsverstehen — Ein wissensbasierter Ansatz*. In: *IT-Schwerpunktheft Fahrerassistenzsysteme*, Seiten 5 – 16, 2007.

[ILT10] ILTER COORDINATING COMMITTEE: *International Long Term Ecological Research (ILTER)*. WWW: `http://www.ilternet.edu`, Dezember 2010. (17.12.2010).

[Inm02] INMON, WILLIAM HARVEY: *Building the Data Warehouse*. Wiley Computer Publishing, 3. Auflage, 2002.

[Int92] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *Information Technology — Database Language SQL*. Technischer Bericht, International Organization for Standardization (ISO), Juli 1992.

[ISO96] ISO/IEC — JOINT TECHNICAL COMMITTEE 1: *ISO/IEC 14977 : 1996(E) — Extended Backus-Naur Form (EBNF)*, 1996.

[Jen11] JENA DEVELOPMENT TEAM: *Joseki - A SPARQL Server for Jena*. WWW: `http://www.joseki.org`, Januar 2011. (02.01.2011).

[Jos08] JOSUTTIS, NICOLAI: *SOA in der Praxis: System-Design für verteilte Geschäftsprozesse*. Dpunkt Verlag, Januar 2008.

[Kar53] KARNAUGH, MAURICE: *The Map Method for Synthesis of Combinational Logic Circuits*. In: *Transactions of the American Institute of Electrical Engineers*, Band 72, Seiten 593 – 599, November 1953.

[KE06] KEMPER, ALFONS und ANDRÉ EICKLER: *Datenbanksysteme – Eine Einführung*. Oldenbourg Verlag, 6. Auflage, 2006.

[KES+07] KUPFER, ANDREAS, SILKE ECKSTEIN, BRITTA STÖRMANN, KARL NEUMANN und BRIGITTE MATHIAK: *Methods for a Synchronised Evolution of Databases and Associated Ontologies*. In: *Databases and Information Systems IV*, Band 155 der Reihe *Frontiers in Artificial Intelligence and Applications*, Seiten 89 – 102. IOS Press, 2007.

[Köh03] KÖHLER, JACOB: *SEMEDA (Semantic Meta-Database): Ontology Based Semantic Integration of Biological Databases*. Doktorarbeit, Technische Fakultät der

Universität Bielefeld, 2003.

[KHFL11] KÖSTER, FRANK, MARCO HANNIBAL, TOBIAS FRANKIEWICZ und KARSTEN LEMMER: *Anwendungsplattform Intelligente Mobilität — Dienstespektrum und Architektur*. In: *Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*, Seiten 11 – 25, Februar 2011.

[Kin10] KING, NICK: *Global Biodiversity Information Facility*. WWW: `http://www.gbif.org`, Dezember 2010. (17.12.2010).

[Kip01] KIPP, MICHAEL: *ANVIL — A Generic Annotation Tool for Multimodal Dialogue*. In: *7th European Conference on Speech Communication and Technology (Eurospeech)*, Seiten 1367 – 1370, 2001.

[KKD08] KRIEGER, HANS-ULRICH, BERND KIEFER und THIERRY DECLERCK: *A Framework for Temporal Representation and Reasoning in Business Intelligence Applications*. In: HINKELMANN, KNUT (Herausgeber): *AI Meets Business Rules and Process Management. Papers from AAAI 2008 Spring Symposium*, Band SS-08-01, Seiten 59 – 70. AAAI Press, 2008.

[KKPS01] KAHAN, JOSÉ, MARJA-RIITTA KOIVUNEN, ERIC PRUD'HOMMEAUX und RALPH R. SWICK: *Annotea: An Open RDF Infrastructure for Shared Web Annotations*. In: *10th international conference on World Wide Web (WWW)*, Seiten 623 – 632, Mai 2001.

[Kli08] KLINOV, PAVEL: *Pronto: a Non-Monotonic Probabilistic Description Logic Reasoner*. The Semantic Web: Research and Applications, 5021:822 – 826, 2008.

[KMG01] KIEMLE, STEFAN, ERHARD MIKUSCH und MARKUS GÖHMANN: *The Product Library - A Scalable Long-Term Repository for Earth Observation Products*. In: ESTEC, EUROPEAN SPACE AGENCY PUBLICATIONS DIVISION (Herausgeber): *Data Systems in Aerospace, Nice, 2001*, Band SP-483. European Space Agency, 2001.

[KMU06] KEMPER, HANS-GEORG, WALID MEHANNA und CARSTEN UNGER: *Business Intelligence — Grundlagen und praktische Anwendungen*. Vieweg Verlag, 2. Auflage, Oktober 2006.

[KN08] KÖSTER, FRANK und ULF NOYER: *Leistungsfähiges Datenmanagement als Rückgrat einer menschzentrierten Entwicklung von Automation / Assistenz*. In: *24. VDI/VW-Gemeinschaftstagung — Integrierte Sicherheit und Fahrerassistenzsysteme*, Nummer 24 in *VDI/VW-Gemeinschatstagung*, Seiten 219 – 226. VDI Wissensforum GmbH, Oktober 2008.

[Knu76] KNUTH, DONALD E.: *Big Omicron and big Omega and big Theta*. ACM SIGACT News, 8(2):18 – 24, Mai 1976.

[Koc08] KOCH, SASCHA: *Analytisches Performance Management*. Doktorarbeit, Carl von Ossietzky Universität Oldenburg, 2008.

[KPS+06] KALYANPUR, ADITYA, BIJAN PARSIA, EVREN SIRIN, BERNARDO CUENCA GRAU und JAMES HENDLER: *Swoop: A Web Ontology Editing Browser*. Web Semantics: Science, Services and Agents on the World Wide Web, 4:144 – 153, Juni 2006.

[Krö10] KRÖTZSCH, MARKUS: *Semantic MediaWiki*. WWW: `http://semantic-mediawiki.org`, Dezember 2010. (06.12.2010).

[KS96] KASHYAP, VIPUL und AMIT SHETH: *Semantic and schematic similarities between database objects: a context-based approach*. The International Journal on Very Large Data Bases, 5(4):276 – 304, 1996.

[Kös07] KÖSTER, FRANK: *Datenbasierte Kompetenz- und Verhaltensanalyse — Anwen-*

dungsbeispiele im selbstorganisierten eLearning. Oldenburg Computer Science Series. Oldenburger Verlag für Wirtschaft, Informatik und Recht, April 2007.

[KS08a] KRÜGER, GUIDO und THOMAS STARK: *Handbuch der Java-Programmierung*. Addison-Wesley, 5. Auflage, 2008.

[KS08b] KŘEMEN, PETR und EVREN SIRIN: *SPARQL-DL Implementation Experience*. In: *OWLED — OWL: Experiences and Directions*, April 2008.

[KVH08] KAUPPINEN, TOMI, JARI VÄÄTÄINEN und EERO HYVÖNEN: *Creating and Using Geospatial Ontology Time Series in a Semantic Cultural Heritage Portal*. In: *5th European Semantic Web Conference 2008*, Seiten 110 – 123. Springer-Verlag, Juni 2008.

[Löf12] LÖFFLER, SVEN: *Einsatz von Systemen zur automatisierten Unterstützung wissenschaftlicher Arbeitsschritte auf Basis von Web-Services und die Durchführung von Annotationen für Experimentaldaten*. Bachelorarbeit, Universität Oldenburg, März 2012.

[LG01] LECOINTRE, GUILLAUME und HERVÉ LE GUYADER: *Biosystematik: Alle Organismen im Überblick*. Springer, 2001.

[LHP+10] LIETZ, HOLGER, MATTHIAS J. HENNING, TIBOR PETZOLD, JOSEF F. KREMS und GERD WANIELIK: *A Software Tool for the Visualization and Annotation of Naturalistic Driving Data Stored in Relational Databases*. In: *European Conference on Human Centred Design for Intelligent Transport Systems*, Seiten 427 – 433, April 2010.

[LS05] LINTERN, ROB und MARGARET-ANNE STOREY: *Jambalaya express: on demand knowledge visualization*. In: *8th International Protégé Conference*, 2005.

[Lun03] LUNZE, JAN: *Automatisierungstechnik — Methoden für die Überwachung und Steuerung kontinuierlicher und ereignisdiskreter Systeme*. Oldenbourg, August 2003.

[Lz09] LIU, LING und M. TAMER ÖZSU (Herausgeber): *Ontology*, Encyclopedia of Database Systems. Springer-Verlag, 2009.

[MAD+05] MURRAY, CHUCK, NICOLE ALEXANDER, SOURI DAS, GEORGE EADON und SIVA RAVADA: *Oracle Spatial — Resource Description Framework (RDF), 10g Release 2 (10.2)*. Oracle, Juli 2005.

[Mag90] MAGNUSON, JOHN J.: *Long-Term Ecological Research and the Invisible Present*. BioScience, 40(7):484 – 501, Juli 1990.

[MAYU05] MATONO, AKIYOSHI, TOSHIYUKI AMAGASA, MASATOSHI YOSHIKAWA und SHUNSUKE UEMURA: *A Path-Based relational RDF database*. In: *Australasian Database Conference (ADC)*, Band 39, Seiten 95 – 103, 2005.

[MBHaSGS97] MICHENER, WILLIAM K., JAMES W. BRUNT, JOHN J. HELLY und THOMAS B. KIRCHNER ANDAND SUSAN G. STAFFORD: *Nongeospatial Metadata for the Ecological Sciences*. Ecological Applications, 7(1):330 – 342, Februar 1997.

[MDD+10] MONTENEGRO, SERGIO, FRANK DANNEMANN, LUTZ DITTRICH, BENJAMIN VOGEL, ULF NOYER, JAN GAČNIK, MARCO HANNIBAL, ANDREAS RICHTER und FRANK KÖSTER: $\frac{(SpacecraftBusController)+(AutomotiveECU)}{2} = UltimateController$. In: *Envision 2020 — Erster Workshop zu Zukunft der Entwicklung softwareintensiver, eingebetteter Systeme*, Februar 2010.

[MDG+00] MIKUSCH, EBERHARD, ERHARD DIEDRICH, MARKUS GÖHMANN, STEFAN KIEMLE, CHRISTOPH RECK, RALF REISSIG, KURT SCHMIDT, WILHELM WILDEGGER und MEINHARD WOLFMÜLLER: *Data Information and Management System for the Produc-*

tion, Archiving and Distribution of Earth Observation Products. In: ESA (Herausgeber): *Data Systems in Aerospace*, Band SP-457, Seiten 401 – 406. ESA, 2000.

[Mey03] MEYER, WOLFGANG: *Entwicklung eines Werkzeugs zur Trainerunterstützung beim simulatorbasierten Pilotentraining*. Diplomarbeit, Cal von Ossietzky Universität Oldenburg, Dezember 2003.

[MHK10] MASTERS, JAMES, RALPH HODGSON und PAUL J. KELLER: *QUDT — Quantities, Units, Dimensions and Data Types in OWL and XML*. WWW: `http://www.qudt.org/`, Februar 2010. (26.02.2010).

[Mic06] MICHENER, WILLIAM K.: *Meta-information concepts for ecological data management*. Ecological Informatics, 1(1):3 – 7, Januar 2006.

[Mil98] MILLER, ERIC: *An Introduction to the Resource Description Framework*. D-Lib Magazine, Mai 1998.

[MK11] MISHRA, RAVI BHUSHAN und SANDEEP KUMAR: *Semantic web reasoners and languages*. Artificial Intelligence Review, 35(4):339 – 368, April 2011.

[MKS10] MIRTL, MICHAEL, KINGA KRAUZE und ANDREW SIER: *European Long-Term Ecosystem Research Network — LTER-Europe*. WWW: `http://www.lter-europe.net`, Dezember 2010. (18.12.2010).

[MLA10] MCAFFER, JEFF, JEAN-MICHEL LEMIEUX und CHRIS ANISZCZYK: *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications*. Addison-Wesley, 2010.

[MM04] MANOLA, FRANK und ERIC MILLER: *RDF Primer*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`, Februar 2004. (01.05.2010).

[Mot06] MOTIK, BORIS: *Reasoning in Description Logics using Resolution and Deductive Databases*. Doktorarbeit, Universität Karlsruhe (TH), Januar 2006.

[Mot10] MOTTA, ENRICO: *NeOn Project*. WWW: `http://www.neon-project.org`, Mai 2010. (05.05.2010).

[Moz99] MOZILLA FOUNDATION: *Mozilla Public License, Version 1.1*. WWW: `http://www.mozilla.org/MPL/1.1/`, 1999. (22.03.2014).

[MPSP09] MOTIK, BORIS, PETER F. PATEL-SCHNEIDER und BIJAN PARSIA: *OWL 2 Web Ontology Language — Structural Specification and Functional-Style Syntax*. W3C Recommendation: `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/`, Oktober 2009. (03.01.2011).

[MSH09] MOTIK, BORIS, ROB SHEARER und IAN HORROCKS: *Hypertableau Reasoning for Description Logics*. Journal of Artificial Intelligence Research, 36:165 – 228, Oktober 2009.

[MSS05] MOTIK, BORIS, ULRIKE SATTLER und RUDI STUDER: *Query Answering for OWL-DL with Rules*. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 1:41 – 60, Juli 2005.

[MvH04] MCGUINNESS, DEBORAH L. und FRANK VAN HARMELEN: *OWL Web Ontology Language Overview*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-owl-features-20040210/`, Februar 2004. (08.04.2010).

[myG09] MYGRID TEAM: *Taverna workbench project website*. WWW: `http://taverna.sourceforge.net/`, November 2009. (07.11.2009).

[MZ08] MALINOWSKI, ELZBIETA und ESTEBAN ZIMANYI: *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*. Data-Centric Systems and Applications. Springer, 2008.

[Nag94] NAGEL, HANS-HELLMUT: *A Vision of 'Vision and Language' Comprises Action: An Example from Road Traffic*. In: *Artificial Intelligence Review*, Band 8, Seiten 189 – 214, 1994.

[Nao09] NAONE, ERICA: *Bedeutungsvolle Zahlen*. Technology Review, August 2009. WWW: `http://www.heise.de/tr/artikel/Bedeutungsvolle-Zahlen-276773.html` (10.07.2010).

[Nat04] NATIONAL INFORMATION STANDARDS ORGANIZATION: *Understanding Metadata*. NISO Press, 4733 Bethesda Avenue, Bethesda, MD 20814 USA, 2004.

[NBK09a] NOYER, ULF, DIRK BECKMANN und FRANK KÖSTER: *Semantic annotation of sensor data to support data analysis processes*. In: *Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM)*, 2009.

[NBK09b] NOYER, ULF, DIRK BECKMANN und FRANK KÖSTER: *Semantic technologies and metadata systematisation for evaluating time series in the context of driving experiments*. In: *11th International Protégé Conference*, Seiten 17 – 18, 2009.

[NBK11] NOYER, ULF, DIRK BECKMANN und FRANK KÖSTER: *Semantic Technologies for Describing Measurement Data in Databases*. In: ANTONIOU, GRIGORIS (Herausgeber): *Extended Semantic Web Conference (ESWC)*, Band 6644 der Reihe *LNCS*, Seiten 198 – 211. Springer-Verlag Berlin Heidelberg, 2011.

[Net04] NETWORK INFERENCE: *Ontology and Data Warehousing - How Data Warehousing Investments can deliver continuous ROI by augmentig them with ontologies and inference capabilities*. Technology White Paper, Network Inference, Inc., 2004.

[Nor09] NORTHWESTERN UNIVERSITY: *Video Annotation Tool*. WWW: `http://dewey.at.northwestern.edu/ppad2/documents/help/video.html`, November 2009. (25.11.2009).

[NOT07] NYULAS, CSONGOR, MARTIN O'CONNOR und SAMSON TU: *DataMaster — a Plug-in for Importing Schemas and Data from Relational Databases into Protégé*. In: *10th Intl. Protégé Conference*, 2007.

[Noy11] NOYER, ARNE: *Animation von reaktiven Systemen mit Statecharts in einem Eclipse-basierten Modellierungswerkzeug*. Diplomarbeit, Ostfalia, Hochschule für angewandte Wissenschaften, Wolfenbüttel, November 2011.

[NRR07] NOYER, ULF, MARTIN RUSCHINZIK und JÜRGEN RATAJ: *Tabellenpartitionierung in PostgreSQL*. iX – Magazin für professionelle Informationstechnik, 4 / 2007:141 – 143, April 2007.

[NSU+13] NOYER, ULF, EIKE A. SCHMIDT, FABIAN UTESCH, DANIEL WAIGAND und FRANK KÖSTER: *Betrachtungen zur systematischen Durchführung von Naturalistic Driving Studies*. In: *Der Fahrer im 21. Jahrhundert — Fahrer, Fahrerunterstützung und Bedienbarkeit*, November 2013.

[NSW+09] NOY, NATALYA F., NIGAM H. SHAH, PATRICIA L. WHETZEL, BENJAMIN DAI, MICHAEL DORF, NICHOLAS GRIFFITH, CLEMENT JONQUET, DANIEL L. RUBIN, MARGARET-ANNE STOREY, CHRISTOPHER G. CHUTE und MARK A. MUSEN: *BioPortal: ontologies and integrated data resources at the click of a mouse*. Nucleic Acids Research, 37:W170 – W173, Mai 2009.

[NT03] NIEMEIER, WOLFGANG und SVEN THOMSEN: *GPS in der Unfallforschung*. In:

POSNAV, März 2003.

[NW08] NEUMANN, THOMAS und GERHARD WEIKUM: *RDF-3X: a RISC-style engine for RDF*. In: *VLDB Endowment*, Seiten 647 – 659, 2008.

[OAF+04] OINN, TOM, MATTHEW ADDIS, JUSTIN FERRIS, DARREN MARVIN, MARTIN SENGER, MARK GREENWOOD, TIM CARVER, KEVIN GLOVER, MATTHEW R. POCOCK, ANIL WIPAT und PETER LI: *Taverna: a tool for the composition and enactment of bioinformatics workflows*. Bioinformatics, 20(17):3045 – 3054, Juni 2004.

[Obj01] OBJECT MANAGEMENT GROUP: *Common Warehouse Metamodel (CWM) Specification*, Februar 2001.

[Obj03] OBJECT MANAGEMENT GROUP: *Common Warehouse Metamodel (CWM) Specification*, März 2003.

[Obj05a] OBJECT MANAGEMENT GROUP: *Meta Object Facility (MOF) Specification*. ISO/IEC 19502, Juli 2005.

[Obj05b] OBJECT MANAGEMENT GROUP: *XML Metadata Interchange Specification*. ISO/IEC 19503, Juli 2005.

[Obj12a] OBJECT MANAGEMENT GROUP: *Unified Modeling Language (OMG UML) — Infrastructure*. ISO/IEC 19505-1, April 2012.

[Obj12b] OBJECT MANAGEMENT GROUP: *Unified Modeling Language (OMG UML) — Superstructure*. ISO/IEC 19505-2, April 2012.

[OGC99] OGC, OPEN GEOSPATIAL CONSORTIUM: *Simple Features Specification for SQL (Revision 1.1)*, Mai 1999.

[Ora11] ORACLE CORPORATION: *Oracle SQL Developer*. WWW: `http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html`, Januar 2011. (31.01.2011).

[Org07] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS): *Web Services Business Process Execution Language Version 2.0*. OASIS Standard: `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`, April 2007. (05.03.2013).

[OSG12] OSGI ALLIANCE: *OSGi Core Release 5*. OSGi Specification, März 2012.

[PA04] PAPADOMANOLAKIS, STRATOS und ANASTASIA AILAMAKI: *Workload-Driven Schema Design for Large Scientific Databases*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 27(4):21 – 28, Dezember 2004.

[PFLT11] PILEGGI, SALVATORE F., CARLOS FERNANDEZ-LLATAS und VICENTE TRAVER: *A Semantic Layer for Embedded Sensor Networks*. ARPN Journal of Systems and Software, 1(3):101 – 107, Juni 2011.

[pgA11] PGADMIN DEVELOPMENT TEAM: *pgAdmin: PostgreSQL administration and management tools*. WWW: `http://www.pgadmin.org`, Januar 2011. (31.01.2011).

[PH04] PEDERSEN, SUSANNE und WILHELM HASSELBRING: *Interoperabilität für Informationssysteme im Gesundheitswesen auf Basis medizinischer Standards*. Informatik-Forschung und Entwicklung, 18(3):174 – 188, Januar 2004.

[PLL06] POLLERES, AXEL, HOLGER LAUSEN und RUBÉN LARA: *Semantische Beschreibung von Web Services*. In: *Semantic Web — Wege zur vernetzten Wissensgesellschaft*. Springer, Juni 2006.

[PPP05] PFEIFFER, SILVIA, CONRAD D. PARKER und ANDRE T. PANG: *Specifying time*

intervals in URI queries and fragments of time-based Web resources. IETF Internet-Draft: `http://www.annodex.net/TR/URI_fragments.html`, März 2005. (06.04.2011).

[PS08] PRUD'HOMMEAUX, ERIC und ANDY SEABORNE: *SPARQL Query Language for RDF*. W3C Recommendation: `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`, Januar 2008. (01.05.2010).

[PSHH04] PATEL-SCHNEIDER, PETER F., PATRICK HAYES und IAN HORROCKS: *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/`, Februar 2004. (04.03.2010).

[Ray11] RAYTHEON BBN TECHNOLOGIES: *OntoNotes*. WWW: `http://www.bbn.com/ontonotes/`, Dezember 2011. (29.12.2011).

[Rit10] RITTER, DANIEL: *Mit Ecken und Kanten — NoSQL-Datenbanken für Graphstrukturen*. iX – Magazin für professionelle Informationstechnik, 7:78 – 82, Juli 2010.

[RKT05] RUSSOMANNO, DAVID J., CARTIK R. KOTHARI und OMOJU A. THOMAS: *Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models*. In: *The 2005 International Conference on Artificial Intelligence*, Seiten 637 – 643, 2005.

[Rot96] ROTHENBERG, JEFF: *Metadata to Support Data Quality and Longevity*. In: *Proceedings of the 1st IEEE Metadata Conference*, Seiten 16 – 18, Juni 1996.

[RQZ07] RUPP, CHRIS, STEFAN QUEINS und BARBARA ZENGLER: *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. Carl Hanser Verlag, 3. Auflage, August 2007.

[SBBK07] SCHAFFERT, SEBASTIAN, FRANCOIS BRY, JOACHIM BAUMEISTER und MALTE KIESEL: *Semantic Wiki*. Informatik-Spektrum, 30:434 – 439, 2007.

[Sch09a] SCHIESSL, CAROLINE: *Modellierung von Belastung und Beanspruchung im Fahrkontext*. Doktorarbeit, Technische Universität Braunschweig, 2009.

[Sch09b] SCHLAUCH, TOBIAS: *DataFinder*. WWW: `http://www.dlr.de/sc/datafinder`, November 2009. (10.11.2009).

[Sch10] SCHULZE, ERNST-DETLEF: *Biodiversity Explorations Information System (BExIS)*. WWW: `http://exploratories.bgc-jena.mpg.de`, Dezember 2010. (18.12.2010).

[SH10] SERAFINI, LUCIANO und MARTIN HOMOLA: *Modular Knowledge Representation and Reasoning in the Semantic Web*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 8, Seiten 147 – 181. Springer, 2010.

[Sha48] SHANNON, CLAUDE ELWOOD: *A mathematical theory of communication*. The Bell System Technical Journal, 27:379 – 423, 1948.

[SHC95] SOMOGYI, ZOLTAN, FERGUS HENDERSON und THOMAS CONWAY: *Mercury: an efficient purely declarative logic programming language*. In: *Proceedings of the Australian Computer Science Conference*, Seiten 499 – 512, Februar 1995.

[SHCO95] SOMOGYI, ZOLTAN, FERGUS HENDERSON, THOMAS CONWAY und RICHARD O'KEEFE: *Logic programming for the real world*. In: *Proceedings of the ILPS '95 Postconference Workshop on Visions for the Future of Logic Programming*, Dezember 1995.

[SHG+10] SCHRÖDER, MARK, MARCO HANNIBAL, JAN GAČNIK, FRANK KÖSTER, CHRISTIAN HARMS und TOBIAS KNOSTMANN: *Ein Labor zur modellbasierten Gestaltung in-*

teraktiver Assistenz und Automation im Automotive-Umfeld. In: *AAET 2010*, Februar 2010.

[SHK03] SCHROETER, RONALD, JANE HUNTER und DOUGLAS KOSOVIC: *Vannotea — A Collaborative Video Indexing, Annotation and Discussion System For Broadband Networks*. In: *Knowledge Markup and Semantic Annotation Workshop*, 2003.

[SHN+11] SCHINDLER, JULIAN, CHRISTIAN HARMS, ULF NOYER, ANDREAS RICHTER, FRANK FLEMISCH, FRANK KÖSTER, THIERRY BELLET, PIERRE MAYENOBE und DOMINIQUE GRUYER: *JDVE: A Joint Driver-Vehicle-Environment Simulation Platform for the Development and Accelerated Testing of Automotive Assistance and Automation Systems*. In: *Human Modelling in Assisted Transportation*, Seiten 233 – 240, 2011.

[SLM+99] STEPHENSON, ARTHUR G., LIA S. LAPIANA, DANIEL R. MULVILLE, PETER J. RUTLEDGE, FRANK H. BAUER, DAVID FOLTA, GREG A. DUKEMAN, ROBERT SACKHEIM und PETER NORVIG: *Mars Climate Orbiter Mishap Investigation Board Phase I Report*. WWW: `ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf`, November 1999. (22.02.2014).

[SMR99] STÖHR, THOMAS, ROBERT MÜLLER und ERHARD RAHM: *An Integrative and Uniform Model for Metadata Management in Data Warehousing Environments*. In: *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, Juni 1999.

[Sou08] SOURCEFORGE.NET: *Jena — A Semantic Web Framework for Java*. WWW: `http://jena.sourceforge.net`, Oktober 2008. (10.10.2008).

[SP05] SCHUPPERT, ANDREAS und RAINER PERNE: *Data Mining mit Prozessdaten*. at – Automatisierungstechnik, 53(7):342–349, Juli 2005.

[SP07] SIRIN, EVRIN und BIJAN PARSIA: *SPARQL-DL: SPARQL Query for OWL-DL*. In: *OWLED — OWL: Experiences and Directions*, Juni 2007.

[SR96] SELIGMAN, LEN und ARNON ROSENTHAL: *A Metadata Resource to Promote Data Integration*. In: *Proc. of IEEE Metadata Conference*, 1996.

[SR13] STAIR, RALPH M. und GEORGE W. REYNOLDS: *Fundamentals of Information Systems*. Cengage Learning, 7. Auflage, 2013.

[SS07] SCHLAUCH, TOBIAS und ANDREAS SCHREIBER: *DataFinder — A Scientific Data Management Solution*. In: *Long-Term Preservation and Value Adding*, Oberfaffenhofen, Oktober 2007.

[SS09] STAAB, STEFFEN und RUDI STUDER (Herausgeber): *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2. Auflage, 2009.

[Sta09] STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH: *The Protégé Ontology Editor and Knowledge Acquisition System*. WWW: `http://protege.stanford.edu/`, April 2009. (16.04.2009).

[SVV99] STAUDT, MARTIN, ANCA VADUVA und THOMAS VETTERLI: *The Role of Metadata for Data Warehousing*. Technischer Bericht, Institut für Informatik, Universität Zürich, 1999.

[SW08] SCHENK, JASON R. und ROBERT L. WADE: *Robotic Systems Technical and Operational Metrics Correlation*. In: *PerMIS'08*, August 2008.

[SWM04] SMITH, MICHAEL K., CHRIS WELTY und DEBORAH L. MCGUINNESS: *OWL Web Ontology Language Guide*. W3C Recommendation: `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`, Februar 2004. (01.05.2010).

[SZG+07] SCHREINER, CHRIS, HARRY ZHANG, CLAUDIA GUERRERO, KARI TORKKOLA und KESHU ZHANG: *A Semi-Automatic Data Annotation Tool for Driving Simulator Data Reduction*. In: *Driving Simulator Conference*, 2007.

[TAJ+10] TÜRKER, CAN, FUAT AKAL, DIETER JOHO, CHRISTIAN PANSE, SIMON BARKOW-OESTERREICHER, HUBERT REHRAUER und RALPH SCHLAPBACH: *B-Fabric: the Swiss Army Knife for life sciences*. In: *Proceedings of the 13th International Conference on Extending Database Technology*, Seiten 717 – 720, 2010.

[TAM+03] TAUTZ, DIETHARD, PETER ARCTANDER, ALESSANDRO MINELLI, RICHARD H. THOMAS und ALFRIED P. VOGLER: *A plea for DNA taxonomy*. Trends in Ecology and Evolution, 18(2):70 – 74, Februar 2003.

[Tan07] TAN, WANG-CHIEW: *Provenance in Databases: Past, Current, and Future*. Bulletin of the Technical Committee on Data Engineering, 30(4):3 – 12, Dezember 2007.

[Ten04] TENZER, AXEL: *Die Fahrerleistungsdatenbank der Volkswagen AG als Werkzeug zur Beobachtung von Fahrerverhalten*. In: *Integrierte Sicherheit und Fahrerassistenzsysteme*, Nummer 1864 in *VDI-Berichte*. VDI-Gesellschaft, Fahrzeug- und Verkehrstechnik, Oktober 2004.

[TH06] TSARKOV, DMITRY und IAN HORROCKS: *FaCT++ Description Logic Reasoner: System Description*. In: *Automated Reasoning*, Seiten 292 – 297, 2006.

[THOD09] TUSCH, GUENTER, XIAOHUI HUANG, MARTIN O'CONNOR und AMAR DAS: *Exploring Microarray Time Series with Protégé*. In: *11th International Protégé Conference*, Seiten 73 – 75, 2009.

[TP07] TÜRKER, CAN und CHRISTIAN PANSE: *The B-Fabric Approach to Data Management for Life Sciences Research at FGCZ*. In: *Third Vital-IT Annual Symposium*. Functional Genomics Center Zurich, September 2007.

[TRKH08] TSERENDORJ, TUVSHINTUR, SEBASTIAN RUDOLPH, MARKUS KRÖTZSCH und PASCAL HITZLER: *Approximate OWL-Reasoning with Screech*. In: *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems*, Seiten 165 – 180. Springer-Verlag, 2008.

[TSGZ06] TORKKOLA, KARI, CHRIS SCHREINER, MIKE GARDNER und KESHU ZHANG: *Development of a Semi-Automatic Data Annotation Tool for Driving Data*. In: *Intelligent Transportation Systems, 2006. Proceedings. 2006 IEEE*, Intelligent Transportation Systems, Seiten 642– 646, September 2006.

[TW94] TUSCHIK, HANS-PETER und HELMUT WOLTER: *Mathematische Logik — kurzgefasst: Grundlagen, Modelltheorie, Entscheidbarkeit, Mengenlehre*. BI-Wissenschaftsverlag, 1994.

[UG96] USCHOLD, MIKE und MICHAEL GRUNINGER: *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review, 11:93 – 136, Juni 1996.

[UKM+10] URBANI, JACOPO, SPYROS KOTOULAS, JASON MAASSEN, FRANK VAN HARMELEN und HENRI BAL: *OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples*. In: *Extended Semantic Web Conference*, 2010.

[Vei52] VEITCH, EDWARD W.: *A chart method for simplifying truth functions*. In: *ACM Annual Conference*, Seiten 127 – 133, 1952.

[Vel10] VELEGRAKIS, YANNIS: *Relational Technologies, Metadata and RDF*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 4, Seiten 41 – 66. Springer, 2010.

[vH01] HEUVEN, VINCENT VAN: *Praat, a system for doing phonetics by computer*. Glot

International, 5(9/10):341 – 345, 2001.

[VNGP10] VIRGILIO, ROBERTO DE, PIERLUIGI DEL NOSTRO, GIORGIO GIANFORME und STEFANO PAOLOZZI: *A Metamodel Approach to Semantic Web Data Management*. In: *Semantic Web Information Management — A Model-Based Perspective*, Kapitel 5, Seiten 67 – 91. Springer, 2010.

[VSA+05] VOLLRATH, M., C. SCHIESSL, T. ALTMÜLLER, M. DAMBIER und C. KORNBLUM: *Erkennung von Fahrmanövern als Indikator für die Belastung des Fahrers*. In: *Fahrer im 21. Jahrhundert*, Band 1919 der Reihe *VDI-Berichte*, Seiten 103 – 112. VDI-Verlag GmbH, 2005.

[W3C09] W3C OWL WORKING GROUP: *OWL 2 Web Ontology Language Document Overview*. W3C Recommendation: `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`, Oktober 2009. (08.04.2010).

[WCL+05] WEERAWARANA, SANJIVA, FRANCISCO CURBERA, FRANK LEYMANN, TONY STOREY und DONALD F. FERGUSON: *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, April 2005.

[WFM06] WELTY, CHRIS, RICHARD FIKES und SELENE MAKARIOS: *A Reusable Ontology for Fluents in OWL*. In: *Fourth International Conference on Formal Ontology in Information Systems (FOIS)*, Seiten 226 – 236, 2006.

[WHA+09] WANG, X., D. HUANG, I. AKTURK, M. BALMAN, G. ALLEN und T. KOSAR: *Semantic Enabled Metadata Management in Petashare*. In: *International Journal of Grid and Utility Computing (IJGUC)*, 2009.

[Wic08] WICKENS, CHRISTOPHER D.: *Multiple Resources and Mental Workload*. Human Factors: The Journal of the Human Factors and Ergonomics Society, 50(3):449 – 455, Juni 2008.

[Wil03] WILTSCHKO, THOMAS: *Mikroskopische Unfallanalyse zur Identifikation von Wirkungsfeldern zukünftiger Fahrerassistenzsysteme*. In: *19. Verkehrswissenschaftliche Tage*, September 2003.

[WK09] WANG, XINQI und TEVFIK KOSAR: *Design and Implementation of Metadata System in PetaShare*. In: *International Conference on Scientific and Statistical Database Management (SSDBM)*, Juni 2009.

[WKB08] WEISS, CATHRIN, PANAGIOTIS KARRAS und ABRAHAM BERNSTEIN: *Hexastore: Sextuple Indexing for Semantic Web Data Management*. In: *VLDB Endowment*, Band 1, Seiten 1008 – 1019, August 2008.

[WLLB06] WEITHÖNER, TIMO, THORSTEN LIEBIG, MARKO LUTHER und SEBASTIAN BÖHM: *What's Wrong with OWL Benchmarks?* In: *Scalable Semantic Web Knowledge Base Systems*, 2006.

[WPR+12] WEISCHEDEL, RALPH, SAMEER PRADHAN, LANCE RAMSHAW, JEFF KAUFMAN, MICHELLE FRANCHINI, MOHAMMED EL-BACHOUTI, NIANWEN XUE, MARTHA PALMER, JENA D. HWANG, CLAIRE BONIAL, JINHO CHOI, AOUS MANSOURI, MAHA FOSTER, ABDEL AATI HAWWARY, MITCHELL MARCUS, ANN TAYLOR, CRAIG GREENBERG, EDUARD HOVY, ROBERT BELVIN und ANN HOUSTON: *OntoNotes Release 5.0*, September 2012.

[WS97] WOODRUFF, ALLISON und MICHAEL STONEBRAKER: *Supporting Fine-grained Data Lineage in a Database Visualization Environment*. In: *Proceedings of the Thirteenth International Conference on Data Engineering*, 1997.

[WSKR03] WILKINSON, KEVIN, CRAIG SAYERS, HARUMI KUNO und DAVE REYNOLDS: *Efficient RDF Storage and Retrieval in Jena2*. In: *First International Workshop on Semantic Web and Databases*, September 2003.

[ZH08] ZHANG, WEISHAN und KLAUS MARIUS HANSEN: *Towards Self-managed Pervasive Middleware using OWL/SWRL ontologies*. In: *Workshop Modeling and Reasoning in Context*, 2008.

D Anhang

D.1 Formale Semantik von RDF

Die folgende Beschreibung der formalen Semantik von RDF erfolgt nach [HKR08, S. 94 – 99] (vgl.a. [Hay04, Kap. 1 u. 3]). Eine in Abschnitt 2.3.1.4 beschriebene *einfache Interpretation* I auf einem Vokabular $\mathcal{V}$ besteht aus den folgenden Elementen (vgl. Abb. 2-12):

- nichtleere Menge R der *Ressourcen* (Domäne oder Universum von $\mathcal{I}$)
- Menge P der *Properties* von $\mathcal{I}$
- Funktion $\mathrm{I_{EXT}}\colon P \to 2^{R \times R}$ (*Extension* des Properties p)
- Funktion $\mathrm{I_S}\colon \mathcal{V} \to R \cup P$
- Funktion $\mathrm{I_L}\colon \mathcal{V}_{getypt} \to R$ mit den getypten Literalen $\mathcal{V}_{getypt} \subset \mathcal{V}$
- Teilmenge $LV \subset R$ enthält ungetypte Literale aus $\mathcal{V}$

Basierend auf der Definition von I wird eine Interpretationsfunktion $\cdot^{\mathcal{I}}$ eingeführt, welche die im „Vokabular $\mathcal{V}$ enthaltenen Literale und URIs auf Ressourcen und Properties abbildet:

- jedes ungetypte Literal $''a''$ wird auf a abgebildet: $(''a'')^{\mathcal{I}} = a$,
- jedes ungetypte Literal mit Sprachangabe $''a''@t$ wird auf das Paar $\langle a, t\rangle$ abgebildet: $(''a''@t)^{\mathcal{I}} = \langle a, t\rangle$,
- jedes getypte Literal l wird auf $\mathrm{I_L}(l)$ abgebildet: $l^{\mathcal{I}} = \mathrm{I_L}(l)$ und
- jede URI u wird auf $\mathrm{I_S}(u)$ abgebildet: $u^{\mathcal{I}} = \mathrm{I_S}(u)$."[HKR08, S. 95] (vgl.a. [Hay04, Kap. 1.4])

Eine RDF-Interpretation baut auf der zuvor eingeführten einfachen Interpretation auf und führt eine spezielle Behandlung für das RDF-Vokabular $\mathcal{V}_{RDF}$ ein.

$$\begin{aligned}\mathcal{V}_{RDF} = \{&\texttt{rdf:type},\ \texttt{rdf:Property},\ \texttt{rdf:XMLLiteral},\ \texttt{rdf:nil},\\ &\texttt{rdf:List},\ \texttt{rdf:Statement},\ \texttt{rdf:subject},\ \texttt{rdf:predicate},\\ &\texttt{rdf:object},\ \texttt{rdf:first},\ \texttt{rdf:rest},\ \texttt{rdf:Seq},\ \texttt{rdf:Bag},\\ &\texttt{rdf:Alt},\ \texttt{rdf:\_1},\ \texttt{rdf:\_2},\ \ldots\}\end{aligned}$$

Somit ist eine *RDF-Interpretation* für ein Vokabular $\mathcal{V}$ „eine einfache Interpretation für das Vokabular $\mathcal{V} \cup \mathcal{V}_{RDF}$, welche zusätzlich folgende Bedingungen erfüllt:

- $x \in P$ genau dann, wenn $\langle x, \texttt{rdf:Property}^{\mathcal{I}}\rangle \in \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$ [. . .]
- wenn $''s''\hat{}\texttt{rdf:XMLLiteral}$ in V enthalten und s ein wohlgeformtes XML-Literal ist, dann
  - $\mathrm{I_L}(''s''\hat{}\texttt{rdf:XMLLiteral})$ ist der XML-Wert [..] von s ;
  - $\mathrm{I_L}(''s''\hat{}\texttt{rdf:XMLLiteral}) \in LV$;
  - $\langle \mathrm{I_L}(''s''\hat{}\texttt{rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}}\rangle \in \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$
- wenn $''s''\hat{}\texttt{rdf:XMLLiteral}$ in $\mathcal{V}$ enthalten und s ein *nicht* wohlgeformtes XML-Literal ist, dann
  - $\mathrm{I_L}(''s''\hat{}\texttt{rdf:XMLLiteral}) \notin LV$ und

– $\langle \mathrm{I_L}(''s''\hat{}\texttt{rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}} \rangle \notin \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$"[HKR08, S. 98 f.] (vgl.a. [Hay04, Kap. 3])

D.2 Liste der axiomatischen RDF-Tripel

Es folgt eine Auflistung der axiomatischen RDF-Tripel nach [HKR08, S. 99] (vgl.a. [Hay04, Kap. 3]), welche von jeder RDF-Implementierung als wahr ausgewertet werden müssen und in Abschnitt 2.3.1 beschrieben sind.

```
rdf:type        rdf:type  rdf:Property .
rdf:subject     rdf:type  rdf:Property .
rdf:predicate   rdf:type  rdf:Property .
rdf:object      rdf:type  rdf:Property .
rdf:first       rdf:type  rdf:Property .
rdf:rest        rdf:type  rdf:Property .
rdf:value       rdf:type  rdf:Property .
rdf:_1          rdf:type  rdf:Property .
rdf:_2          rdf:type  rdf:Property .
...
rdf:nil         rdf:type  rdf:List .
```

D.3 Formale Semantik von RDFS

Die folgende Beschreibung der formalen Semantik von RDFS erfolgt nach [HKR08, S. 99 ff.] (vgl.a. [Hay04, Kap. 4]). Eine *RDFS-Interpretation* nach Abschnitt 2.3.2.3 stellt eine Erweiterung einer RDF-Interpretation dar, die um die Eigenschaften zur Beschreibung von terminologischem Wissen erweitert ist und hierfür das RDFS-Vokabular $\mathcal{V}_{RDFS}$ zur Verfügung stellt:

$$\begin{aligned}\mathcal{V}_{RDFS} = \{&\texttt{rdfs:domain}, \texttt{rdfs:range}, \texttt{rdfs:Resource}, \texttt{rdfs:Literal},\\ &\texttt{rdfs:Datatype}, \texttt{rdfs:Class}, \texttt{rdfs:subClassOf},\\ &\texttt{rdfs:subPropertyOf}, \texttt{rdfs:member}, \texttt{rdfs:Container},\\ &\texttt{rdfs:ContainerMembershipProperty}, \texttt{rdfs:comment},\\ &\texttt{rdfs:seeAlso}, \texttt{rdfs:isDefinedBy}, \texttt{rdfs:label}\}\end{aligned}$$

Zur Beschreibung der zusätzlichen Bedingungen wird eine neue Hilfsfunktion $\mathrm{I_{CEXT}}\colon R \to 2^R$ eingeführt, welche die *(Klassen-)Extension* von y genannt wird. Weiterhin wird eine neue Menge C als Extension der speziellen RDFS-URI `rdfs:Class` eingeführt. Eine *RDFS-Interpretation* für ein Vokabular $\mathcal{V}$ ist somit eine RDF-Interpretation des Vokabulars $\mathcal{V} \cup \mathcal{V}_{RDFS}$, welche nach [HKR08, S. 100] bzw. [Hay04, Kap. 4] zusätzlich die folgenden Kriterien erfüllt:

- $x \in \mathrm{I_{CEXT}}(y)$ genau dann, wenn $\langle x, y \rangle \in \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$
  - $C = \mathrm{I_{CEXT}}(\texttt{rdfs:Class}^{\mathcal{I}})$
  - $R = \mathrm{I_{CEXT}}(\texttt{rdfs:Resource}^{\mathcal{I}})$
  - $LV = \mathrm{I_{CEXT}}(\texttt{rdfs:Literal}^{\mathcal{I}})$
- „Wenn $\langle x, y \rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:domain}^{\mathcal{I}})$ und $\langle u, v \rangle \in \mathrm{I_{EXT}}(x)$, dann $u \in \mathrm{I_{CEXT}}(y)$ [...]
- Wenn $\langle x, y \rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:range}^{\mathcal{I}})$ und $\langle u, v \rangle \in \mathrm{I_{EXT}}(x)$, dann $v \in \mathrm{I_{CEXT}}(y)$ [...]

- $\mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$ ist reflexiv und transitiv auf P [...]
- Wenn $\langle x, y\rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$, dann $x, y \in P$ und $\mathrm{I}_{\mathrm{EXT}}(x) \subseteq \mathrm{I}_{\mathrm{EXT}}(y)$ [...]
- Wenn $x \in C$, dann $\langle x, \texttt{rdfs:Resource}^{\mathcal{I}}\rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$ [...]
- Wenn $\langle x, y\rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$, dann $x, y \in C$ und $\mathrm{I}_{\mathrm{CEXT}}(x) \subseteq \mathrm{I}_{\mathrm{CEXT}}(y)$ [...]
- $\mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$ ist reflexiv und transitiv auf C. [...]
- Wenn $x \in \mathrm{I}_{\mathrm{CEXT}}(\texttt{rdfs:ContainerMembershipProperty}^{\mathcal{I}})$, dann $\langle x, \texttt{rdfs:member}^{\mathcal{I}}\rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$. [...]
- Wenn $x \in \mathrm{I}_{\mathrm{CEXT}}(\texttt{rdfs:Datatype}^{\mathcal{I}})$, dann $\langle x, \texttt{rdfs:Literal}^{\mathcal{I}}\rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$"[HKR08, S. 100 f.] (vgl.a. [Hay04, Kap. 4])

D.4 Liste der axiomatischen RDFS-Tripel

Es folgt eine Auflistung der axiomatischen RDFS-Tripel nach [HKR08, S. 101 ff.] (vgl.a. [Hay04, Kap. 4]), welche von jeder RDFS-Implementierung als wahr ausgewertet werden müssen und in Abschnitt 2.3.2 beschrieben sind.

```
rdf:type            rdfs:domain  rdfs:Resource .
rdfs:domain         rdfs:domain  rdf:Property .
rdfs:range          rdfs:domain  rdf:Property .
rdfs:subPropertyOf  rdfs:domain  rdf:Property .
rdfs:subClassOf     rdfs:domain  rdfs:Class .
rdf:subject         rdfs:domain  rdf:Statement .
rdf:predicate       rdfs:domain  rdf:Statement .
rdf:object          rdfs:domain  rdf:Statement .
rdfs:member         rdfs:domain  rdfs:Resource .
rdf:first           rdfs:domain  rdf:List .
rdf:rest            rdfs:domain  rdf:List .
rdfs:seeAlso        rdfs:domain  rdfs:Resource .
rdfs:isDefinedBy    rdfs:domain  rdfs:Resource .
rdfs:comment        rdfs:domain  rdfs:Resource .
rdfs:label          rdfs:domain  rdfs:Resource .
rdf:value           rdfs:domain  rdfs:Resource .

rdf:type            rdfs:range  rdfs:Class .
rdfs:domain         rdfs:range  rdfs:Class .
rdfs:range          rdfs:range  rdfs:Class .
rdfs:subPropertyOf  rdfs:range  rdf:Property .
rdfs:subClassOf     rdfs:range  rdfs:Class .
rdf:subject         rdfs:range  rdfs:Resource .
rdf:predicate       rdfs:range  rdfs:Resource .
rdf:object          rdfs:range  rdfs:Resource .
rdfs:member         rdfs:range  rdfs:Resource .
rdf:first           rdfs:range  rdfs:Resource .
rdf:rest            rdfs:range  rdf:List .
rdfs:seeAlso        rdfs:range  rdfs:Resource .
rdfs:isDefinedBy    rdfs:range  rdfs:Resource .
rdfs:comment        rdfs:range  rdfs:Literal .
rdfs:label          rdfs:range  rdfs:Literal .
rdf:value           rdfs:range  rdfs:Resource .
```

```
rdfs:ContainerMembershipProperty
                  rdfs:subClassOf     rdf:Property .
rdf:Alt           rdfs:subClassOf     rdfs:Container .
rdf:Bag           rdfs:subClassOf     rdfs:Container .
rdf:Seq           rdfs:subClassOf     rdfs:Container .

rdfs:isDefinedBy  rdfs:subPropertyOf  rdfs:seeAlso .

rdf:XMLLiteral    rdf:type            rdfs:Datatype .
rdf:XMLLiteral    rdfs:subClassOf     rdfs:Literal .
rdfs:Datatype     rdfs:subClassOf     rdfs:Class .

rdf:_1  rdf:type
        rdfs:ContainerMembershipProperty .
rdf:_1  rdfs:domain  rdfs:Resource .
rdf:_1  rdfs:range   rdfs:Resource .
rdf:_2  rdf:type
        rdfs:ContainerMembershipProperty .
rdf:_2  rdfs:domain  rdfs:Resource .
rdf:_2  rdfs:range   rdfs:Resource .
...
```

D.5 Übersicht der RDF(S)-Sprachelemente

Es folgt eine Auflistung sämtlicher in RDF(S) zulässiger Sprachelemente nach [HKR08, S. 85 f.] (vgl.a. [BG04]), welche in Abschnitt 2.3.1 und 2.3.2 beschrieben sind.

RDF(S) Klassen

```
rdfs:Class     rdf:Property    rdfs:Resource  rdfs:Literal
rdfs:Datatype  rdf:XMLLiteral
```

RDF(S) Eigenschaften

```
rdfs:range          rdfs:domain  rdf:type      rdfs:subClassOf
rdfs:subPropertyOf  rdfs:label   rdfs:comment
```

RDF(S) Listen

```
rdfs:Container                    rdf:Bag      rdf:Seq   rdf:Alt
rdf:li                            rdf:_1       rdf:_2    ...
rdfs:ContainerMembershipProperty  rdfs:member  rdf:List  rdf:first
rdf:rest                          rdf:nil
```

Reifikation

```
rdf:Statement  rdf:subject  rdf:predicate  rdf:object
```

RDF Attribute

`rdf:about` `rdf:ID` `rdf:resource` `rdf:nodeID`
`rdf:datatype`

XML Attribute

`xml:base` `xml:ns` `xml:lang`

RDF(S) weitere Attribute

`rdf:RDF` `rdfs:seeAlso` `rdfs:isDefinedBy` `rdf:value`

D.6 Überblick über die OWL-Teilsprachen

Es folgt ein Überblick über die in Abschnitt 2.3.3.3 beschriebenen „Teilsprachen von OWL und ihre wichtigsten Eigenschaften. [...]

- OWL Full:
  - enthält OWL DL und OWL Lite,
  - enthält als einzige OWL-Teilsprache ganz RDFS,
  - sehr ausdrucksstark,
  - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind,
  - unentscheidbar,
  - wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt [, . . .]
  - [höchste Laufzeitkomplexität.]
- OWL DL:
  - enthält OWL Lite und ist Teilsprache von OWL Full,
  - entscheidbar,
  - wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt, [. . .]
  - [mittlere Laufzeitkomplexität.]
- OWL Lite:
  - ist Teilsprache von OWL DL und OWL Full,
  - entscheidbar,
  - weniger ausdrucksstark, [. . .]
  - [geringste Laufzeitkomplexität.]“[HKR08, S.127] (vgl.a. [MvH04, Kap. 1.3])

D.7 Übersicht der Beziehungen und Eigenschaften von OWL-Properties

Gegeben seien die Properties $r, s \in P$ und die Individuen $a, b, c \in R$. Dann lassen sich die in Abschnitt 2.3.3.2 beschriebenen Property-Beziehungen und Property-Eigenschaften in einer OWL-Ontologie nach [BvHH+04, Kap. 4] wie folgt formal beschreiben:

- Beziehungen zwischen zwei Rollen
  - Sei r `owl:equivalentProperty` s, dann gilt: $a\ r\ b. \Leftrightarrow a\ s\ b.$
  - Sei r `owl:inverseOf` s, dann gilt: $a\ r\ b. \Leftrightarrow b\ s\ a.$
- Eigenschaften einer Rolle
  - Sei r *transitiv*, dann gilt: $a\ r\ b. \wedge b\ r\ c. \Rightarrow a\ r\ c.$
  - Sei r *symmetrisch*, dann gilt: $a\ r\ b. \Rightarrow b\ r\ a.$
  - Sei r *funktional*, dann gilt: $a\ r\ b. \wedge a\ r\ c. \Rightarrow b$ `owl:sameAs` c
  - Sei r *invers funktional*, dann gilt: $a\ r\ c. \wedge b\ r\ c. \Rightarrow a$ `owl:sameAs` b

D.8 Übersicht der OWL-Sprachelemente

Es folgt eine Auflistung sämtlicher in OWL zulässiger Sprachelemente nach [HKR08, S. 156 f.] (vgl.a. [PSHH04]), welche in Abschnitt 2.3.3 beschrieben sind. Elemente, die in OWL Lite nicht oder nur eingeschränkt verfügbar sind, sind mit $\star$ markiert.

Kopf einer Ontologie

`rdfs:comment`
`rdfs:seeAlso`
`owl:versionInfo`
`owl:backwardCompatibleWith`
`owl:DeprecatedClass`
`owl:imports`
`rdfs:label`
`rdfs:isDefinedBy`
`owl:priorVersion`
`owl:incompatibleWith`
`owl:DeprecatedProperty`

Beziehungen zwischen Individuen

`owl:sameAs`
`owl:AllDifferent` zusammen mit `owl:distinctMembers`
`owl:differentFrom`

Klassenkonstruktoren und Beziehungen
owl:Class
owl:Nothing
owl:disjointWith⋆
owl:intersectionOf
owl:complementOf⋆
owl:Thing
rdfs:subClassOf
owl:equivalentClass
owl:unionOf⋆

Rollenrestriktionen mit owl:Restriction **und** owl:Property
owl:allValuesFrom
owl:cardinality⋆
owl:oneOf⋆, für Datentypen zusammen mit owl:DataRange⋆
owl:someValuesFrom
owl:minCardinality⋆
owl:hasValue
owl:maxCardinality⋆

Rollenkonstruktoren, -beziehungen und -eigenschaften
owl:ObjectProperty
rdfs:subPropertyOf
rdfs:domain
owl:DatatypeProperty
owl:equivalentProperty
owl:inverseOf
rdfs:range
rdf:resource=''&owl;TransitiveProperty''
rdf:resource=''&owl;SymmetricProperty''
rdf:resource=''&owl;FunctionalProperty''
rdf:resource=''&owl;InverseFunctionalProperty''

D.9 Übersicht der Einschränkungen von OWL DL

Die Einschränkungen ergeben sich aus der im folgenden zitierten Aufstellung: „Es folgt eine Liste von Anforderungen, die erfüllt sein müssen, damit eine OWL-Ontologie in OWL DL liegt:

- Einschränkung von RDF(S): Es dürfen nur Sprachelemente von RDF(S) verwendet werden, die explizit für OWL DL erlaubt sind. [...] Insbesondere ist die Verwendung von rdfs:Class und rdf:Property verboten.
- Typentrennung und -deklaration: Es muss klar zwischen Individuen, Klassen, abstrakten Rollen, konkreten Rollen, Datentypen und den im Kopf zu deklarierenden Ontologieeigenschaften unterschieden werden. [...]

- Einschränkungen für konkrete Rollen: Die Rolleneigenschaften `owl:inverseOf`, `owl:TransitiveProperty`, `owl:SymmetricProperty` und `owl:InverseFunctionalProperty` dürfen konkreten Rollen nicht zugewiesen werden.
- Einschränkungen für abstrakte Rollen: Zahlenrestriktionen mittels `owl:cardinality`, `owl:minCardinality` oder `owl:maxCardinality` dürfen nicht mit transitiven Rollen, deren Inversen oder deren Oberrollen verwendet werden."[HKR08, S. 153] (vgl.a. [BvHH+04, Kap. 8.2])

D.10 Übersicht der Einschränkungen von OWL Lite

Die Einschränkungen ergeben sich aus der im folgenden zitierten Aufstellung: „Für OWL Lite gelten folgende Einschränkungen im Vergleich zu OWL Full:

- Alle Einschränkungen für OWL DL gelten auch für OWL Lite.
- Eingeschränkte Klassenkonstruktoren: Verboten ist die Verwendung von `owl:oneOf`, `owl:unionOf`, `owl:complementOf`, `owl:hasValue`, `owl:disjointWith` und `owl:DataRange`.
- Eingeschränkte Zahlenrestriktionen: Zahlenrestriktionen sind nur mit den Werten 0 und 1 erlaubt.
- [. . .] In manchen Situationen ist die Verwendung von Klassennamen vorgeschrieben: im Subjekt von `owl:equivalentClass` und `rdfs:subClassOf`, im Objekt von `rdfs:domain`.
- [. . .] In manchen Situationen ist die Verwendung von Klassennamen oder Rollenrestriktionen vorgeschrieben: Im Objekt von `owl:equivalentClass`, `rdfs:subClassOf` und `rdf:type`. Außerdem kann `owl:intersectionOf` nur auf Klassennamen und Rollenrestriktionen angewendet werden.
- [. . .] In manchen Situationen ist die Verwendung von Klassennamen oder Rollenrestriktionen vorgeschrieben: im Objekt von `owl:allValuesFrom`, `owl:someValuesFrom` und `rdfs:range`."[HKR08, S. 153 f.] (vgl.a. [BvHH+04, Kap. 8.3])

D.11 Datentypen aus XML-Schema

Abbildung D-1 stellt die Datentypen von XML-Schema dar, welche nach Abschnitt 2.3.1.2 auch für RDF(S) und OWL verfügbar sind. Der einfache Basisdatentyp (anySimpleType) wird hierbei in verschiedene vordefinierte Basisdatentypen (built-in primitive types) unterteilt. Für die Basisdatentypen *string* und *decimal* existieren dann noch weitere Spezialisierungen in Form von vordefinierten abgeleiteten Datentypen (built-in derived types).

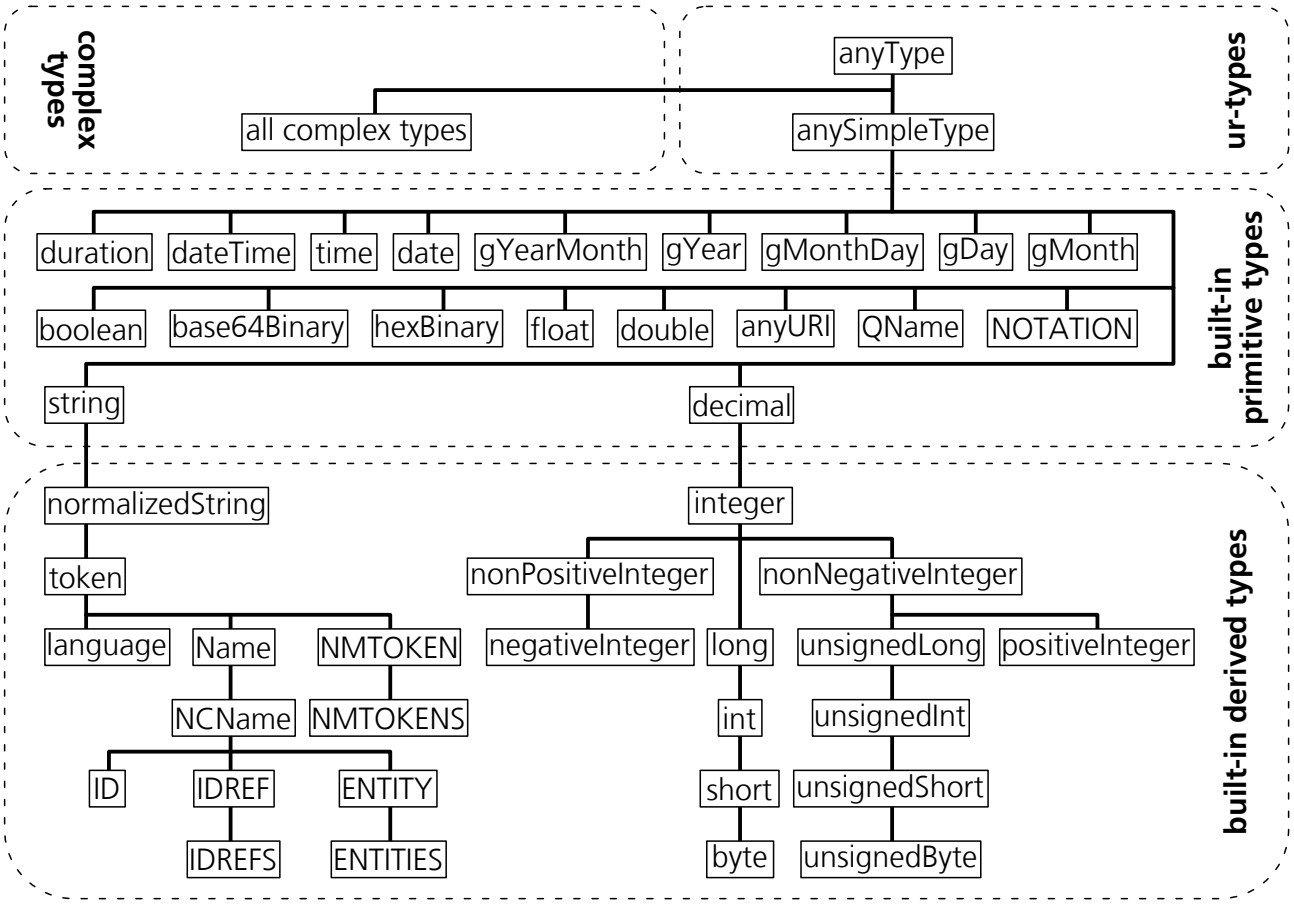


Abbildung D-1: Hierarchie der XML-Schema-Datentypen[2] (nach [BPM04]), die auch in RDF(S) und OWL verfügbar sind

D.12 Beispiel für logische Klassenkonstruktoren

In Listing D-1 wird das Beispiel aus Abschnitt 2.3.3.4 zu logischen Klassenkonstruktoren in Turtle-Schreibweise dargestellt.

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<Blinken>
    a owl:Class .

<Schulterblick>
    a owl:Class .

<Geradeausfahren>
    a owl:Class .

<Abbiegen>
    a owl:Class ;
    owl:unionOf (
        [
            a owl:Class ;
            owl:intersectionOf (
                <Blinken>
                <Schulterblick>
            )
        ]
        [
            a owl:Class ;
            owl:intersectionOf (
                <Blinken>
                [
                    a owl:Class ;
                    owl:complementOf <Geradeausfahren>
                ]
            )
        ]
        [
            a owl:Class ;
            owl:intersectionOf (
                <Schulterblick>
                [
                    a owl:Class ;
                    owl:complementOf <Geradeausfahren>
                ]
            )
        ]
    ) .
```

Listing D-1: Darstellung des Beispiels aus Abschnitt 2.3.3.4 in Turtle-Notation

D.13 Relationen zwischen Zeitintervallen

Es folgt in Tab. D-1 eine Aufstellung der nach Abschnitt 2.3.7.1 möglichen Relationen, in welchen Zeitintervalle zueinander stehen können. Eine ausführliche Behandlung von Zeitpunkten, Zeitintervallen und deren Relationen zueinander liegt in [All83] vor.

Relation	Symbol	Symbol für Inverses	Bildliches Beispiel
X bevor Y (before)	<	>	XXX YYY
X gleich Y (equal)	=	=	XXX YYY
X trifft Y (meets)	m	mi	XXXYYY
X überlappt Y (overlaps)	o	oi	XXX YYY
X während Y (during)	d	di	XXX YYYYYY
X startet Y (starts)	s	si	XXX YYYYY
X beendet Y (finishes)	f	fi	XXX YYYYY

Tabelle D-1: Mögliche Relationen zwischen zwei Zeitintervallen X und Y [All83]

D.14 Vollständiges Automaten-Beispiel

Der Automat $A = (Q, \Sigma, \delta, Strasse\_Folgen)$ nach Abschnitt 3.5.2 in Abb. 3-17 besitzt die im Folgenden aufgeführten Eigenschaften:

$$
\begin{aligned}
A &= (Q, \Sigma, \delta, Strasse\_Folgen) \\
Q &= \{Strasse\_Folgen, Kreuzung\_Fahren, Abbiegen\} \\
\Sigma &= \{Kreuzung, \neg Kreuzung, Lenken, \neg Lenken\} \\
\delta &= \{ \\
&\quad (Strasse\_Folgen, Kreuzung, Kreuzung\_Fahren), \\
&\quad (Kreuzung\_Fahren, \neg Kreuzung, Strasse\_Folgen), \\
&\quad (Kreuzung\_Fahren, Lenken, Abbiegen), \\
&\quad (Abbiegen, \neg Lenken, Strasse\_Folgen), \\
&\quad (Strasse\_Folgen, Lenken, Strasse\_Folgen), \\
&\quad (Strasse\_Folgen, \neg Lenken, Strasse\_Folgen), \\
&\quad (Strasse\_Folgen, \neg Kreuzung, Strasse\_Folgen), \\
&\quad (Kreuzung\_Fahren, Kreuzung, Kreuzung\_Fahren), \\
&\quad (Kreuzung\_Fahren, \neg Lenken, Kreuzung\_Fahren), \\
&\quad (Abbiegen, Lenken, Abbiegen), \\
&\quad (Abbiegen, Kreuzung, Abbiegen), \\
&\quad (Abbiegen, \neg Kreuzung, Abbiegen) \\
\}
\end{aligned}
$$

Das Listing D-2 stellt den vollständigen Automaten aus Abb. 3-17 in Turtle-Notation dar.

```
1  @prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
2  @prefix owl:     <http://www.w3.org/2002/07/owl#> .
3  @prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
4
5  <Ereignis>
6        a       owl:Class .
7
8  <Zeile>
9        a       owl:Class .
10
11 <Zustand>
12       a       owl:Class .
13
14 <e>   a       owl:ObjectProperty ;
15       rdfs:domain <Zeile> ;
16       rdfs:range <Ereignis> .
17
18 <n>   a       owl:ObjectProperty ;
19       rdfs:domain <Zeile> ;
20       rdfs:range <Zeile> .
21
22 <z>   a       owl:ObjectProperty ;
23       rdfs:domain <Zeile> ;
24       rdfs:range <Zustand> .
25
26 <Abbiegen>
27       a       <Zustand> .
28
29 <Kreuzung_fahren>
30       a       <Zustand> .
31
32 <Strasse_folgen>
```

```
33        a       <Zustand> .
34
35  <Kreuzung>
36        a       <Ereignis> .
37
38  <neKreuzung>
39        a       <Ereignis> .
40
41  <Lenken>
42        a       <Ereignis> .
43
44  <neLenken>
45        a       <Ereignis> .
46
47  <Zeile1>
48        a       <Zeile> ;
49        <n>     <Zeile10> ;
50        <z>     <Strasse_folgen> .
51
52  <Zeile10>
53        a       <Zeile> ;
54        <e>     <Kreuzung> ;
55        <n>     <Zeile20> .
56
57  <Zeile20>
58        a       <Zeile> ;
59        <e>     <Lenken> ;
60        <n>     <Zeile30> .
61
62  <Zeile30>
63        a       <Zeile> ;
64        <e>     <neLenken> ;
65        <n>     <Zeile40> .
66
67  <Zeile40>
68        a       <Zeile> ;
69        <e>     <neKreuzung> .
70
71  <x1>  a       swrl:Variable .
72
73  <x2>  a       swrl:Variable .
74
75  <Transition1>
76        a       swrl:Imp ;
77        swrl:body ([ a       swrl:IndividualPropertyAtom ;
78                     swrl:argument1 <x1> ;
79                     swrl:argument2 <Strasse_folgen> ;
80                     swrl:propertyPredicate
81                            <z>
82                   ] [ a       swrl:IndividualPropertyAtom ;
83                     swrl:argument1 <x1> ;
84                     swrl:argument2 <x2> ;
85                     swrl:propertyPredicate
86                            <n>
87                   ] [ a       swrl:IndividualPropertyAtom ;
88                     swrl:argument1 <x2> ;
89                     swrl:argument2 <Kreuzung> ;
90                     swrl:propertyPredicate
91                            <e>
92                   ]) ;
93        swrl:head ([ a       swrl:IndividualPropertyAtom ;
94                     swrl:argument1 <x2> ;
95                     swrl:argument2 <Kreuzung_fahren> ;
96                     swrl:propertyPredicate
97                            <z>
98                   ]) .
99
100 <Transition2>
101       a       swrl:Imp ;
102       swrl:body ([ a       swrl:IndividualPropertyAtom ;
103                    swrl:argument1 <x1> ;
104                    swrl:argument2 <Kreuzung_fahren> ;
105                    swrl:propertyPredicate
106                           <z>
107                  ] [ a       swrl:IndividualPropertyAtom ;
108                    swrl:argument1 <x1> ;
```

```
                    swrl:argument2 <x2> ;
                    swrl:propertyPredicate
                            <n>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <neKreuzung> ;
                    swrl:propertyPredicate
                            <e>
                  ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ]) .

<Transition3>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <Kreuzung_fahren> ;
                    swrl:propertyPredicate
                            <z>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <x2> ;
                    swrl:propertyPredicate
                            <n>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Lenken> ;
                    swrl:propertyPredicate
                            <e>
                  ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Abbiegen> ;
                    swrl:propertyPredicate
                            <z>
                  ]) .

<Transition4>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <Abbiegen> ;
                    swrl:propertyPredicate
                            <z>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <x2> ;
                    swrl:propertyPredicate
                            <n>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <neLenken> ;
                    swrl:propertyPredicate
                            <e>
                  ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ]) .

<StdTrans1>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <x2> ;
```

```
                    swrl:propertyPredicate
                            <n>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Lenken> ;
                    swrl:propertyPredicate
                            <e>
                  ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ]) .

<StdTrans2>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <x2> ;
                    swrl:propertyPredicate
                            <n>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <neLenken> ;
                    swrl:propertyPredicate
                            <e>
                  ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ]) .

<StdTrans3>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <x2> ;
                    swrl:propertyPredicate
                            <n>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <neKreuzung> ;
                    swrl:propertyPredicate
                            <e>
                  ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x2> ;
                    swrl:argument2 <Strasse_folgen> ;
                    swrl:propertyPredicate
                            <z>
                  ]) .

<StdTrans4>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <Kreuzung_fahren> ;
                    swrl:propertyPredicate
                            <z>
                  ] [ a       swrl:IndividualPropertyAtom ;
                    swrl:argument1 <x1> ;
                    swrl:argument2 <x2> ;
                    swrl:propertyPredicate
```

```
                        <n>
                ] [ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x2> ;
                  swrl:argument2 <Kreuzung> ;
                  swrl:propertyPredicate
                        <e>
                ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x2> ;
                  swrl:argument2 <Kreuzung_fahren> ;
                  swrl:propertyPredicate
                        <z>
                ]) .

<StdTrans5>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x1> ;
                  swrl:argument2 <Kreuzung_fahren> ;
                  swrl:propertyPredicate
                        <z>
                ] [ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x1> ;
                  swrl:argument2 <x2> ;
                  swrl:propertyPredicate
                        <n>
                ] [ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x2> ;
                  swrl:argument2 <neLenken> ;
                  swrl:propertyPredicate
                        <e>
                ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x2> ;
                  swrl:argument2 <Kreuzung_fahren> ;
                  swrl:propertyPredicate
                        <z>
                ]) .

<StdTrans6>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x1> ;
                  swrl:argument2 <Abbiegen> ;
                  swrl:propertyPredicate
                        <z>
                ] [ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x1> ;
                  swrl:argument2 <x2> ;
                  swrl:propertyPredicate
                        <n>
                ] [ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x2> ;
                  swrl:argument2 <Lenken> ;
                  swrl:propertyPredicate
                        <e>
                ]) ;
      swrl:head ([ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x2> ;
                  swrl:argument2 <Abbiegen> ;
                  swrl:propertyPredicate
                        <z>
                ]) .

<StdTrans7>
      a       swrl:Imp ;
      swrl:body ([ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x1> ;
                  swrl:argument2 <Abbiegen> ;
                  swrl:propertyPredicate
                        <z>
                ] [ a       swrl:IndividualPropertyAtom ;
                  swrl:argument1 <x1> ;
                  swrl:argument2 <x2> ;
                  swrl:propertyPredicate
                        <n>
```

```
337                   ] [ a        swrl:IndividualPropertyAtom ;
338                     swrl:argument1 <x2> ;
339                     swrl:argument2 <Kreuzung> ;
340                     swrl:propertyPredicate
341                             <e>
342                   ]) ;
343         swrl:head ([ a        swrl:IndividualPropertyAtom ;
344                     swrl:argument1 <x2> ;
345                     swrl:argument2 <Abbiegen> ;
346                     swrl:propertyPredicate
347                             <z>
348                   ]) .
349
350 <StdTrans8>
351         a        swrl:Imp ;
352         swrl:body ([ a        swrl:IndividualPropertyAtom ;
353                     swrl:argument1 <x1> ;
354                     swrl:argument2 <Abbiegen> ;
355                     swrl:propertyPredicate
356                             <z>
357                   ] [ a        swrl:IndividualPropertyAtom ;
358                     swrl:argument1 <x1> ;
359                     swrl:argument2 <x2> ;
360                     swrl:propertyPredicate
361                             <n>
362                   ] [ a        swrl:IndividualPropertyAtom ;
363                     swrl:argument1 <x2> ;
364                     swrl:argument2 <neKreuzung> ;
365                     swrl:propertyPredicate
366                             <e>
367                   ]) ;
368         swrl:head ([ a        swrl:IndividualPropertyAtom ;
369                     swrl:argument1 <x2> ;
370                     swrl:argument2 <Abbiegen> ;
371                     swrl:propertyPredicate
372                             <z>
373                   ]) .
```

Listing D-2: Beispiel eines Automaten zu Abb. 3-17 in Turtle-Notation

Abbildung D-2 zeigt die TBox der in Abschnitt 3.5.3.3 eingeführten Zeit-Ontologie, welche für Bereiche in der Belegungsvisualisierung (Abschnitt 3.4.3) eingesetzt wird. Unterklassenbeziehungen sind in der Abbildung mit einem durchgezogenen Pfeil dargestellt. Gestrichelte Pfeile von einer Klasse zu einer anderen kennzeichnen Rollen mit ihrer Domain- und Range-Definition.

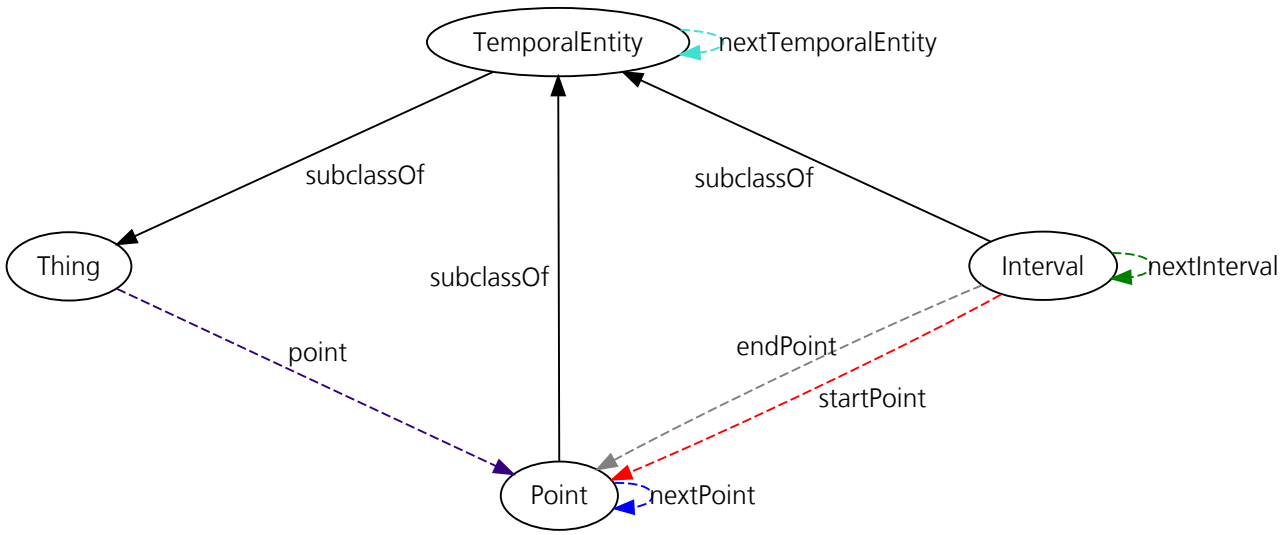


Abbildung D-2: Zeit-Ontologie zur Umsetzung von Bereichen in der Belegungsvisualisierung

D.15 Umsetzung des Semantic Database Browsers

Tabelle D-2 und D-3 zeigen eine Aufstellung zum Umfang des SIMDa-Frameworks und des Semantic Database Browsers, welche in Abschnitt 4 vorgestellt werden. In diesen Tabellen sind die Klassen bzw. Interfaces und Zeilen Quellcode je nach Java-Package aufgeschlüsselt. Die entstandenen Klassen im Tools-Package sind so allgemein gehalten, dass sie nicht nur für semantische Annotationen relevant sind. Bei der Auflistung für den Semantic Database Browser (SDB) werden weiterhin entstandene Klassen berücksichtigt, welche Ontologien erzeugen oder Daten zur Demonstration mit diesen annotieren, sodass diese im Semantic Database Browser betrachtet werden können.

Package	Klassen und Interfaces	Zeilen
simda	26	1.291
simda.jena	22	5.005
de.dlr.fs.tools.database.model	18	3.109
de.dlr.fs.tools.database.model.flavour	3	440
de.dlr.fs.tools.jdbc.csv	18	6.373
de.dlr.fs.tools.jdbc.csv.factories	9	815
de.dlr.fs.tools.swing	12	1.346
de.dlr.fs.tools.swing.graph	17	3.202

Tabelle D-2: Umfang an Java-Quellcode für das SIMDa-Framework, Stand 27.02.2011

Package	Klassen und Interfaces	Zeilen
annotator	3	359
annotator.jena	4	300
annotator.tasks	11	2.383
ontology	11	720
ontology.jena	13	5.458
ontology.jena.builder	4	1.487
ontology.jena.builder.test	7	1.366
semanticdbbrowser	30	6.488
semanticdbbrowser.timeline	4	539

Tabelle D-3: Umfang an Java-Quellcode für den Semantic Database Browser, Stand 27.02.2011

Abbildung D-3 zeigt die Verwaltung von Namensräumen im Semantic Database Browser.

Abbildung D-4 zeigt in den Semantic Database Browser eingebundene Elemente der QUDT-Ontologie, die eine Verbindung zur DBpedia und über diese zur Wikipedia herstellen.

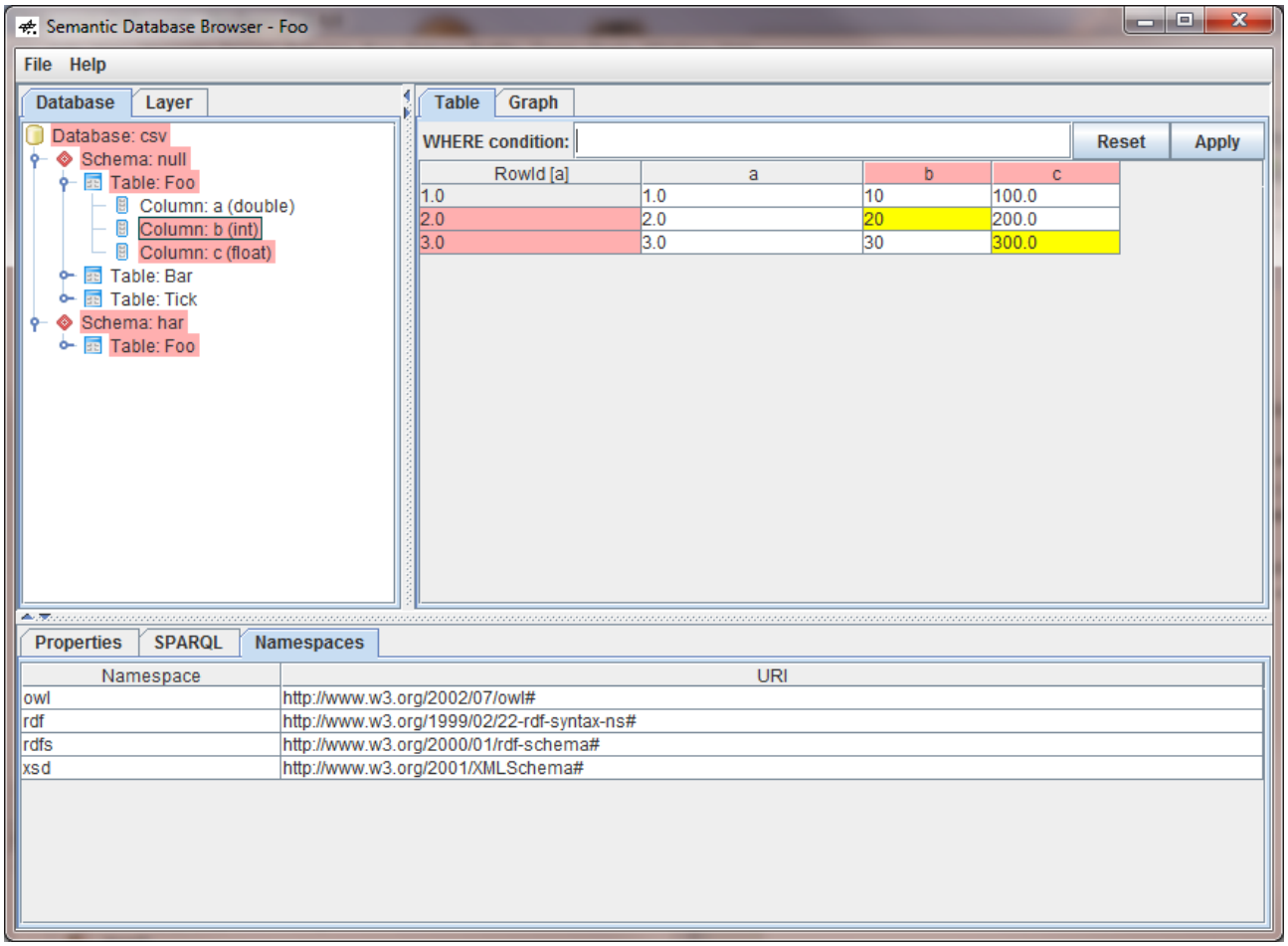


Abbildung D-3: SDB, Verwaltung von Namensräumen

Abbildung D-4: SDB, Einbindung der DBpedia

D.16 Anwendung semantischer Annotationen auf SQL-Ebene

Die Listings D-3 bis D-6 enthalten die notwendigen PL/SQL-Funktionen, um wie in Abschnitt 4.3 gezeigt relationale Daten mit ihren Annotationen gemeinsam relational verarbeiten zu können. Listing D-3 enthält die Funktion `mapdbelement_prefix`, welche nach Abschnitt 3.2.1 den Präfix der URLs für die im Dominion-DataStore enthaltenen Datenbank-Elemente zurückgibt. Die Funktion `mapdbelement` in Listing D-4 übernimmt die Abbildung der Datenbank-Elemente in URIs. Da die eingesetzte Oracle-Datenbank jedes RDF-Modell in einer separaten Tabelle speichert, übernimmt die Funktion `getmodeltable` (Listing D-5) die Abbildung von RDF-Modell zu der entsprechenden Relation.

```
1  CREATE FUNCTION mapdbelement_prefix
2  RETURN VARCHAR2 AS
3  BEGIN
4      RETURN 'http://www.dlr.de/ts/dominion-datastore/db/orcl';
5  END;
```

Listing D-3: PL/SQL-Funktion liefert den URI-Präfix für Datenbank-Elemente

```
1  CREATE FUNCTION mapdbelement (
2      in_schema VARCHAR2 DEFAULT null,
3      in_table VARCHAR2 DEFAULT null,
4      in_column VARCHAR2 DEFAULT null,
5      in_row VARCHAR2 DEFAULT null
6  ) RETURN VARCHAR2 AS
7      prefix VARCHAR2(128);
8  BEGIN
9      prefix := mapdbelement_prefix();
10     IF (in_row IS NOT NULL AND in_column IS NOT NULL) THEN
11         RETURN prefix || '/schema/' || in_schema || '/table/' || in_table ||
12             '/column/' || in_column || '/row/' || in_row;
13     ELSIF (in_row IS NOT NULL) THEN
14         RETURN prefix || '/schema/' || in_schema || '/table/' || in_table ||
15             '/row/' || in_row;
16     ELSIF (in_column IS NOT NULL) THEN
17         RETURN prefix || '/schema/' || in_schema || '/table/' || in_table ||
18         '/column/' || in_column;
19     ELSIF (in_table IS NOT NULL) THEN
20         RETURN prefix || '/schema/' || in_schema || '/table/' || in_table;
21     ELSIF (in_schema IS NOT NULL) THEN
22         RETURN prefix || '/schema/' || in_schema;
23     ELSE
24         RETURN prefix;
25     END IF;
26 END;
```

Listing D-4: PL/SQL-Funktion zur Abbildung von Datenbank-Elementen in URIs

```
1  CREATE FUNCTION getmodeltable (
2      in_modelname VARCHAR2
3  ) RETURN VARCHAR2 AS
4      out_table VARCHAR2(32);
5  BEGIN
6      SELECT short_name INTO out_table
7          FROM MODELS
8          WHERE long_name=in_modelname;
9      RETURN out_table;
10 END;
```

Listing D-5: PL/SQL-Funktion zur Abbildung von RDF-Modellen auf Relationen

Die relationale Darstellung der Annotationen für eine anzugebende Tabelle mit Massendaten übernimmt die Funktion `bulk_annotation` (Listing D-6). Über ein klassisches `JOIN` lassen sich die relational dargestellten Annotationen mit den eigentlichen Massendaten gemeinsam nutzen. Der Rückgabetyp der Funktion ist `ind_bulk_annotation_table` und repräsentiert eine

Tabelle mit Zeilen vom Typ `ind_bulk_annotation`, welche eine Zeile oder Zelle und die Annotation einer Massendatentabelle darstellt. Die Annotation wird im Datentyp `SDO_RDF_TRIPLE_S` gespeichert, welcher jeweils einen vollständigen Aussagetripel enthält. Die Anwendung der Funktion `bulk_annotation` ist in Listing 4-2 dargestellt.

```
1  CREATE OR REPLACE TYPE ind_bulk_annotation AS
2      OBJECT (schema_id VARCHAR2(32), table_id VARCHAR2(32),
3           column_id VARCHAR2(32), row_id NUMBER(10), triple SDO_RDF_TRIPLE_S);
4  CREATE OR REPLACE TYPE ind_bulk_annotation_table AS
5      TABLE OF ind_bulk_annotation;
6
7  CREATE FUNCTION bulk_annotation (
8      in_schema VARCHAR2,
9      in_table VARCHAR2
10 ) RETURN ind_bulk_annotation_table PIPELINED AS
11     -- Temporäre Variable(n)
12     mdl_table VARCHAR2(32);
13     myquery VARCHAR2(1024);
14     -- Variablen zur Cursor-Verarbeitung
15     TYPE ref0 IS REF CURSOR;
16     cur0 ref0;
17     out_rec ind_bulk_annotation :=
18         ind_bulk_annotation(in_schema, in_table, null, null, null);
19 BEGIN
20     mdl_table := getmodeltable(mapdbelement(in_schema, in_table) || '-bulk') || '_tpl';
21
22     -- Hole Zeilen-Annotationen
23     myquery :=
24         ' SELECT ' ||
25             'CAST( ' ||
26                 'substr( ' ||
27                     'a.triple.get_subject(), ' ||
28                     'instr(a.triple.get_subject(), ''/'', 1, 12) + 1, ' ||
29                     'length(a.triple.get_subject()) - ' ||
30                         'instr(a.triple.get_subject(), ''/'', 1, 12) - 1 ' ||
31                 ') ' ||
32             'AS number) AS row_id, ' ||
33             'a.triple ' ||
34         'FROM ' || mdl_table || ' a ' ||
35         'WHERE a.triple.get_subject() ' ||
36         'LIKE ''<' || mapdbelement(in_schema, in_table, null, '%') || ''''';
37
38     OPEN cur0 FOR myquery;
39     LOOP
40         FETCH cur0 INTO out_rec.row_id, out_rec.triple;
41         EXIT WHEN cur0%NOTFOUND;
42         PIPE ROW(out_rec);
43     END LOOP;
44     CLOSE cur0;
45
46     -- Hole Zellen-Annotationen
47     myquery :=
48         ' SELECT ' ||
49             'substr( ' ||
50                 'a.triple.get_subject(), ' ||
51                 'instr(a.triple.get_subject(), ''/'', 1, 12) + 1, ' ||
52                 'instr(a.triple.get_subject(), ''/'', 1, 13) - ' ||
53                     'instr(a.triple.get_subject(), ''/'', 1, 12) - 1 ' ||
54             ') AS column_id, ' ||
55             'CAST( ' ||
56                 'substr( ' ||
57                     'a.triple.get_subject(), ' ||
58                     'instr(a.triple.get_subject(), ''/'', 1, 14) + 1, ' ||
59                     'length(a.triple.get_subject()) - ' ||
60                         'instr(a.triple.get_subject(), ''/'', 1, 14) - 1 ' ||
61                 ') ' ||
62             'AS number) AS row_id, ' ||
63             'a.triple ' ||
64         'FROM ' || mdl_table || ' a ' ||
65         'WHERE a.triple.get_subject() ' ||
66         'LIKE ''<' || mapdbelement(in_schema, in_table, '%', '%') || ''''';
67
68     OPEN cur0 FOR myquery;
69     LOOP
70         FETCH cur0 INTO out_rec.column_id, out_rec.row_id, out_rec.triple;
```

```
71          EXIT WHEN cur0%NOTFOUND;
72          PIPE ROW(out_rec);
73      END LOOP;
74      CLOSE cur0;
75
76      RETURN;
77  END;
```

Listing D-6: PL/SQL-Funktion zur relationalen Darstellung semantischer Annotationen für eine Massendatentabelle

D.17 Einsatz konkreter Ontologien

Listing D-7 zeigt diejenigen Tripel, welche nach Abschnitt 5.1 eine Datenbankspalte über die QUDT-Ontologie mit der DBpedia und der Wikipedia verknüpfen.

```
@prefix dbpedia:    <http://dbpedia.org/resource/> .
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .
@prefix quantity:   <http://data.nasa.gov/qudt/owl/quantity#> .
@prefix qudt:       <http://data.nasa.gov/qudt/owl/qudt#> .
@prefix skos:       <http://www.w3.org/2004/02/skos/core#> .
@prefix unit:       <http://data.nasa.gov/qudt/owl/unit#> .

<http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/u_myuser/
      ↪ table/20100716_105436_expstate/column/velocity>
    qudt:unit unit:MeterPerSecond .

unit:MeterPerSecond
    qudt:quantityKind quantity:LinearVelocity .

quantity:LinearVelocity
    skos:exactMatch dbpedia:Velocity .

dbpedia:Velocity
    foaf:page <http://en.wikipedia.org/wiki/Velocity> .
```

Listing D-7: Verbindung zwischen einer Spalte und der DBpedia über die QUDT-Ontologie

Abbildung D-5 zeigt die TBox, der in Abschnitt 5.2 beschriebenen Datenbank-Ontologie. Unterklassenbeziehungen sind in der Abbildung mit einem durchgezogenen Pfeil dargestellt. Gestrichelte Pfeile von einer Klasse zu einer anderen kennzeichnen Rollen mit ihrer Domain- und Range-Definition. Weiterhin sind die gestrichelt dargestellten Properties alle als Sub-Properties von `containsDatabaseElement` realisiert. Diese Ontologie kann verwendet werden, um Repräsentanten von Datenbank-Elementen (vgl. Abschnitt 3.2.1) in einer ABox als Individuen zu systematisieren.

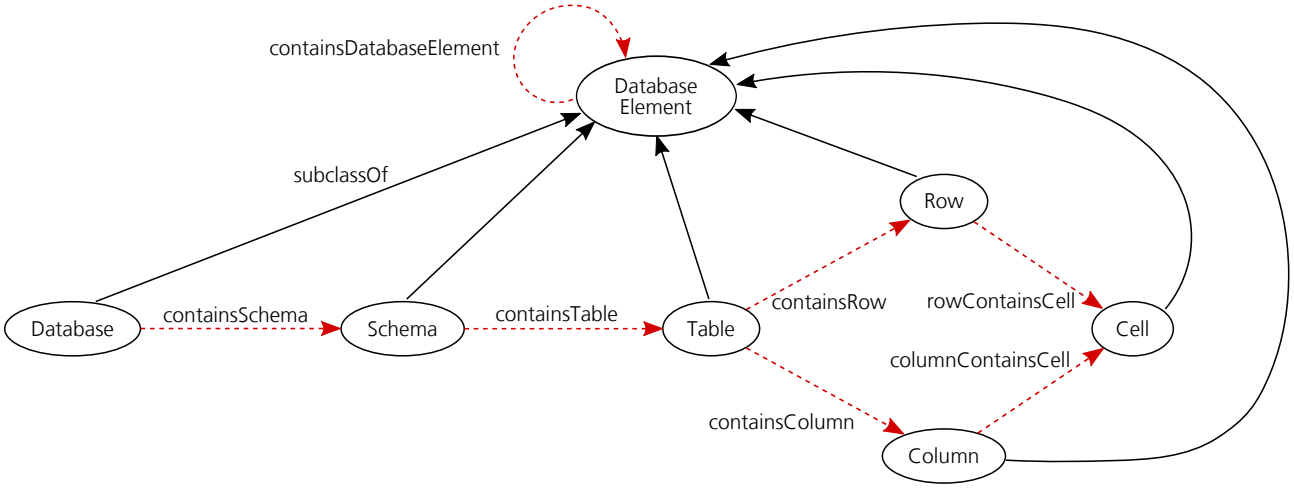


Abbildung D-5: Datenbank-Ontologie zur Beschreibung von semantischen Annotationen

Abweichend zu den in Abschnitt 3.5.3.2 eingeführten Modellierungsdetails wird für die Manöver-Ontologie (Abschnitt 5.3) eine erweiterte Modellierung zur Berücksichtigung vorhergehender Ereignisse durch Reasoning eingesetzt. Diese erweiterte Modellierung der Properties ist in Abb. D-6 dargestellt. Die zusätzlich eingeführten Sub-Properties erlauben es zu erkennen, welche Ereignisse explizit annotiert wurden und welche durch Reasoning von vorhergehenden Annotationen abgeleitet wurden.

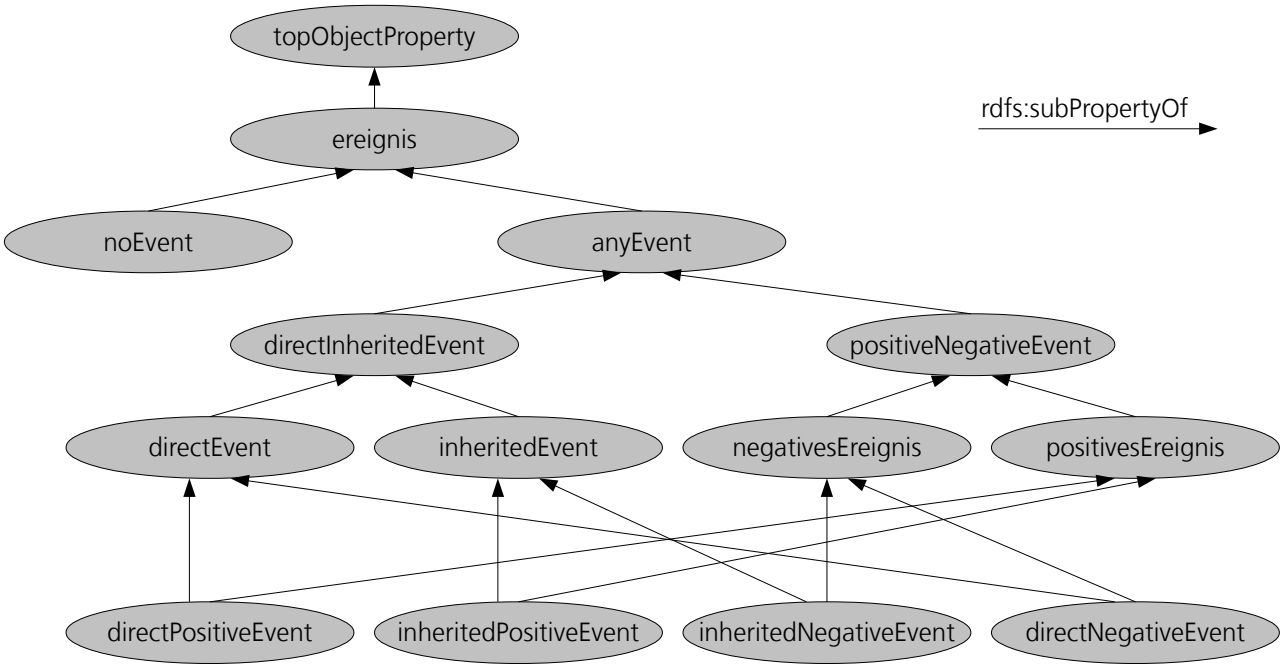


Abbildung D-6: Hierarchie der Properties für die Annotation der Fahrereignisse als Erweiterung der Modellierung von Abschnitt 3.5.3.2

Tabelle D-4 zeigt die benötigten Zeiten für ausgewählte Zeilenanzahlen zur Berechnung der vollständigen Manöver-Ontologie mit 15 Manövern. Sie bildet somit die Grundlage für Abb. 5-3. Als erster Schritt wird die TBox und danach die ABox erstellt. Anschließend erfolgt die eigentliche Berechnung der Ontologie. Als nächster Schritt der Auswertung werden mittels der in Abschnitt 5.3 dargestellten SPARQL-Abfrage genau die Zeilen mit den berechneten Manövern abgerufen. In der Spalte *Statistik* wird zusätzlich die Zeit angegeben, welche benötigt wurde, Statistiken zu der ausgewerteten Ontologie zu erfassen. Diese Statistiken umfassen das Zählen der Klassen, Individuen, Regeln, Manöver etc. Weiterhin sind die dargestellten Zeilen als kumulativ zu verstehen. Daher umfasst beispielsweise die in der Spalte *Statistik* angegebene Zeit die Zeiten sämtlicher vorher durchgeführter Schritte. Tabelle D-5 zeigt die identischen Messungen, wobei die aktualisierten Versionen Pellet 2.3 und Java 1.7 eingesetzt wurden. Für die in Tab. D-6 aufgelisteten Messungen wurde weiterhin die in Abschnitt 5.4.2 beschriebene Vorstufe eingesetzt.

Zeilen	TBox [ms]	ABox [ms]	Berechnung [ms]	SPARQL [ms]	Statistik [ms]
1	68	69	107	111	639
10	36	40	212	216	2.645
25	36	43	819	825	8.180
50	35	48	1.790	1.800	16.535
75	37	55	3.447	3.473	25.563
100	35	59	5.947	5.976	36.004
125	36	66	8.773	8.817	48.523

Tabelle D-4: Zeitdauer zur Berechnung der Ontologie mit 15 Manövern mit einer verschiedenen Anzahl von Zeilen

Die Funktionen aus Abb. 5-4 zum Vergleich verschiedener Komplexitäten sind wie folgt parametrisiert:

- Angenähertes Wachstum: $f(x) = 105 + 1.34364 \cdot x^{1.81743}$

Zeilen	TBox [ms]	ABox [ms]	Berechnung [ms]	SPARQL [ms]	Statistik [ms]
1	125	125	171	343	1.139
10	31	47	203	203	2.325
25	31	31	717	717	5.725
50	31	31	1.591	1.607	13.619
75	31	62	3.167	3.198	21.075
100	31	46	5.023	5.054	29.515
125	32	63	7.644	7.676	38.610

Tabelle D-5: Zeitdauer zur Berechnung der Ontologie mit 15 Manövern mit Pellet 2.3 und Java 7

Zeilen	TBox & ABox [ms]	Vorstufe [ms]	Berechnung [ms]	SPARQL [ms]
1	156	172	234	234
10	110	141	250	266
25	125	156	359	374
50	125	172	765	765
75	141	203	936	952
100	218	312	1.497	1.497
125	156	281	1.529	1.544

Tabelle D-6: Zeitdauer zur Berechnung der Ontologie mit 15 Manövern mit Pellet 2.3, Java 7 und Vorstufe

- Angenähertes Wachstum (P2.3, J7): $g(x) = 140 + 1.67275 \cdot x^{1.73653}$
- Angenähertes Wachstum (Vorstufe): $h(x) = 147.552 + 12.1922 \cdot x$
- Lineares Wachstum: $l(x) = 105 + 11.866 \cdot x$
- Quadratisches Wachstum: $q(x) = 105 + 1.07583 \cdot x^2$

Listing D-8 zeigt das aus dem semantischen Wiki (Abb. 5-6) exportierte RDF-Dokument, welches Annotationen zu dem Manöver *Anhalten Stand* enthält (vgl. Abschnitt 5.3). In den Zeilen 29 bis 39 befinden sich die eigentlichen Nutzdaten, welche im Annotationsbaum (Abb. 5-7) zu dem Manöver dargestellt werden. Alternativ ist es möglich, anstatt ein RDF-Dokument für einen einzelnen Artikel des Wikis zu exportieren, die Annotationen sämtlicher im Wiki vorhandener Artikel in einem umfassenden RDF-Dokument bereitzustellen.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
    <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
    <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
    <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
    <!ENTITY swivt 'http://semantic-mediawiki.org/swivt/1.0#'>
    <!ENTITY wiki 'http://ts.dlr.de/ontology/Spezial:URIResolver/'>
    <!ENTITY property 'http://ts.dlr.de/ontology/Spezial:URIResolver/Attribut-3A'>
    <!ENTITY wikiurl 'http://ts.dlr.de/ontology/'>
]>

<rdf:RDF
    xmlns:rdf="&rdf;"
    xmlns:rdfs="&rdfs;"
    xmlns:owl ="&owl;"
    xmlns:swivt="&swivt;"
    xmlns:wiki="&wiki;"
    xmlns:property="&property;"
    xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <!-- Ontology header -->
    <owl:Ontology rdf:about="&wikiurl;Spezial:ExportRDF/Anhalten_Stand">
        <swivt:creationDate
                rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
            2011-03-15T21:18:54+01:00
        </swivt:creationDate>
        <owl:imports rdf:resource="http://semantic-mediawiki.org/swivt/1.0" />
    </owl:Ontology>
    <!-- exported page data -->
    <swivt:Subject rdf:about="http://ts.dlr.de/ontology/Anhalten_Stand">
        <rdfs:label>Anhalten Stand</rdfs:label>
        <swivt:page rdf:resource="&wikiurl;Anhalten_Stand"/>
        <rdfs:isDefinedBy
            rdf:resource="&wikiurl;Spezial:ExportRDF/Anhalten_Stand"/>
        <rdf:type rdf:resource="http://ts.dlr.de/ontology/Maneuver"/>
        <foaf:homepage rdf:resource="http://elib.dlr.de/43697/"/>
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            initiales Manöver
        </rdfs:comment>
    </swivt:Subject>
    <!-- auxiliary definitions -->
    <owl:DatatypeProperty
            rdf:about="http://www.w3.org/2000/01/rdf-schema#comment">
        <rdfs:label>Hat Kommentar</rdfs:label>
        <swivt:page rdf:resource="&wikiurl;Attribut:Hat_Kommentar"/>
        <rdfs:isDefinedBy
            rdf:resource="&wikiurl;Spezial:ExportRDF/Attribut:Hat_Kommentar"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:about="http://xmlns.com/foaf/0.1/homepage">
        <rdfs:label>Hat Homepage</rdfs:label>
        <swivt:page rdf:resource="&wikiurl;Attribut:Hat_Homepage"/>
        <rdfs:isDefinedBy
            rdf:resource="&wikiurl;Spezial:ExportRDF/Attribut:Hat_Homepage"/>
    </owl:ObjectProperty>
    <owl:Class rdf:about="http://ts.dlr.de/ontology/Maneuver">
        <rdfs:label>Manöver</rdfs:label>
        <swivt:page rdf:resource="&wikiurl;Kategorie:Man%C3%B6ver"/>
        <rdfs:isDefinedBy
            rdf:resource="&wikiurl;Spezial:ExportRDF/Kategorie:Man%C3%B6ver"/>
    </owl:Class>
    <!-- References to the SWiVT Ontology,
        see http://semantic-mediawiki.org/swivt/ -->
    <owl:AnnotationProperty rdf:about="&swivt;page">
        <rdfs:isDefinedBy rdf:resource="http://semantic-mediawiki.org/swivt/1.0"/>
    </owl:AnnotationProperty>
    <owl:AnnotationProperty rdf:about="&swivt;creationDate">
        <rdfs:isDefinedBy rdf:resource="http://semantic-mediawiki.org/swivt/1.0"/>
    </owl:AnnotationProperty>
    <owl:Class rdf:about="&swivt;Subject">
        <rdfs:isDefinedBy rdf:resource="http://semantic-mediawiki.org/swivt/1.0"/>
    </owl:Class>
    <!-- Created by Semantic MediaWiki, http://semantic-mediawiki.org -->
</rdf:RDF>
```

Listing D-8: RDF-Beschreibung des Manövers *Anhalten Stand* aus dem semantischen Wiki

ISSN 1866-721X